Projet SDD

Yuxiang ZHANG et Antoine LECOMTE
Groupe 2

Mai 2025

1 Apprentissage supervisé

1.1 Classifier Perceptron

1.1.1 Classification binaire avec Perceptron - Version 1

Temps total d'entraînement : 6.202062129974365 secondes

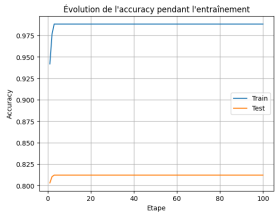

Figure 1: Accuracy of Perceptron Version 1

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]

Taux moyen de bonne classification : 0.9875

Écart-type : 0.0250

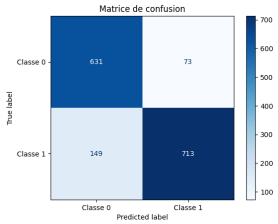

Figure 2: Confusion Matrix of Perceptron Version 1

1.1.2 Classification binaire avec Bag-of-Words (Binaire) avec Perceptron - Version 2

Temps total d'entraînement : 3.705735921859741 secondes

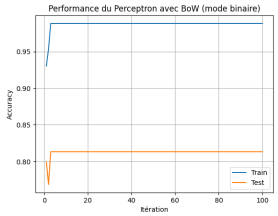

Figure 3: Accuracy of Perceptron Version 2

Accuracy finale (train): 0.9883720930232558

Accuracy finale (test) : 0.8132930513595166

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375

- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]
Taux moyen de bonne classification : 0.9875
Écart-type : 0.0250

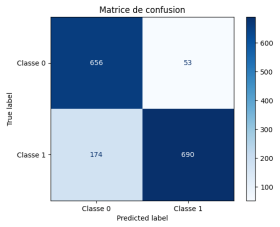

Figure 4: Confusion Matrix of Perceptron Version 2

1.1.3 Classification binaire avec Comptage de mots (Bag-of-Words count) avec Perceptron - Version 3

Temps total d'entraînement : 3.550997018814087 secondes

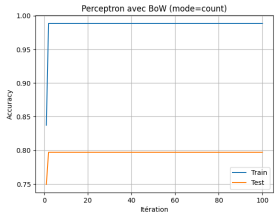

Figure 5: Accuracy of Perceptron Version 3

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.7969788519637462

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8125
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375

Taux de bonne classification par fold : [np.float64(1.0), np.float64(0.9375), np.float64(1.0), np.float64(0.8125), np.float64(0.9375)]
Taux moyen de bonne classification : 0.9375
Écart-type : 0.0685

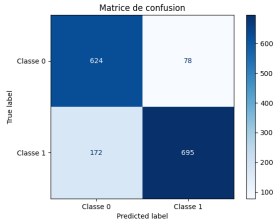

Figure 6: Confusion Matrix of Perceptron Version 3

1.1.4 Classification binaire avec Fréquence relative (Bag-of-Words freq) avec Perceptron - Version 4

Temps total d'entraînement : 3.531724214553833 secondes

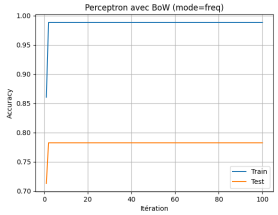

Figure 7: Accuracy of Perceptron Version 4

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.7824773413897281

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750

- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8125
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.7500
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375

Taux de bonne classification par fold : [np.float64(1.0), np.float64(0.875), np.float64(0.8125), np.float64(0.75), np.float64(0.9375)]
Taux moyen de bonne classification : 0.8750
Écart-type : 0.0884

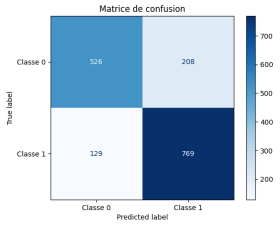

Figure 8: Confusion Matrix of Perceptron Version 4

1.1.5 Classification binaire avec TF-IDF avec Perceptron - Version 5

Temps total d'entraînement : 3.6801087856292725 secondes

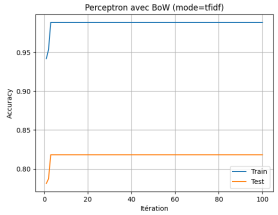

Figure 9: Accuracy of Perceptron Version 5

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.8181268882175227

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750

- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(0.875), np.float64(0.9375), np.float64(1.0), np.float64(0.875), np.float64(1.0)]
Taux moyen de bonne classification : 0.9375
Écart-type : 0.0559

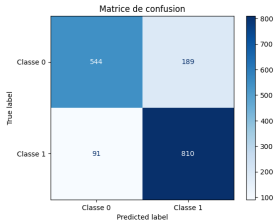

Figure 10: Confusion Matrix of Perceptron Version 5

1.1.6 Classification multi-classe avec Perceptron

Temps total d'entraînement : 14.799072980880737 secondes
Accuracy (multi-classe): 0.4550

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8125
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(0.9375), np.float64(0.875), np.float64(0.8125), np.float64(0.875), np.float64(1.0)]
Taux moyen de bonne classification : 0.9000
Écart-type : 0.0637

1.2 Classifier PerceptronBiais

1.2.1 Classification binaire avec PerceptronBiais - Version 1

Temps total d'entraînement : 6.455108880996704 secondes

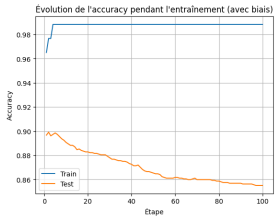

Figure 11: Accuracy of PerceptronBiais Version 1

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.8549848942598187

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]
Taux moyen de bonne classification : 0.9875
Écart-type : 0.0250

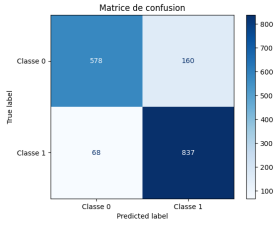

Figure 12: Confusion Matrix of PerceptronBiais Version 1

1.2.2 Classification binaire avec Bag-of-Words (Binaire) avec PerceptronBiais - Version 2

Temps total d'entraînement : 3.687659978866577 secondes

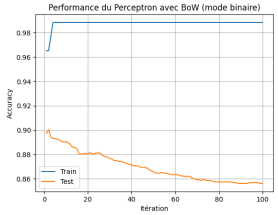

Figure 13: Accuracy of PerceptronBiais Version 2

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.8561933534743202

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]
Taux moyen de bonne classification : 0.9875

Écart-type : 0.0250

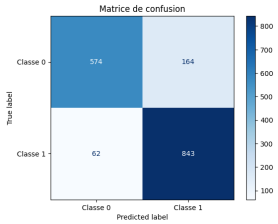

Figure 14: Confusion Matrix of PerceptronBiais Version 2

1.2.3 Classification binaire avec Comptage de mots (Bag-of-Words count) avec PerceptronBiais - Version 3

Temps total d'entraînement : 3.669743061065674 secondes

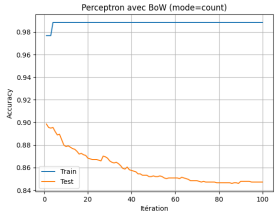

Figure 15: Accuracy of PerceptronBiais Version 3

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.8471299093655589

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]

Taux moyen de bonne classification : 0.9875

Écart-type : 0.0250

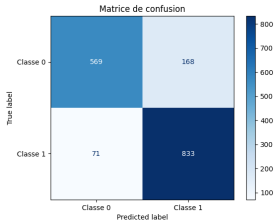

Figure 16: Confusion Matrix of PerceptronBiais Version 3

1.2.4 Classification binaire avec Fréquence relative (Bag-of-Words freq) avec PerceptronBiais - Version 4

Temps total d'entraînement : 4.125259876251221 secondes

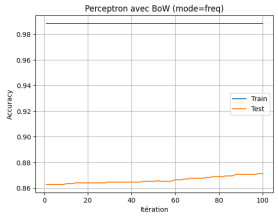

Figure 17: Accuracy of PerceptronBiais Version 4

Accuracy finale (train): 0.9883720930232558

Accuracy finale (test) : 0.8712990936555891

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375

- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]
Taux moyen de bonne classification : 0.9875
Écart-type : 0.0250

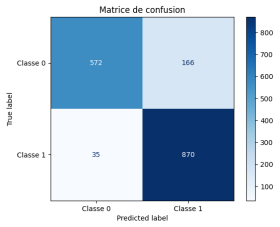

Figure 18: Confusion Matrix of PerceptronBiais Version 4

1.2.5 Classification binaire avec TF-IDF avec PerceptronBiais - Version 5

Temps total d'entraînement : 4.1352858543396 secondes

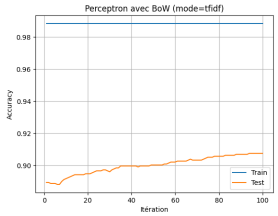

Figure 19: Accuracy of PerceptronBiais Version 5

Accuracy finale (train): 0.9883720930232558
Accuracy finale (test) : 0.9075528700906345

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(1.0), np.float64(1.0), np.float64(1.0), np.float64(0.9375), np.float64(1.0)]
Taux moyen de bonne classification : 0.9875
Écart-type : 0.0250

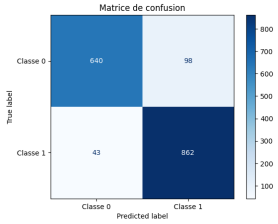

Figure 20: Confusion Matrix of PerceptronBiais Version 5

1.2.6 Classification multi-classe avec PerceptronBiais

Accuracy (multi-classe avec PerceptronBiais et PCA): 0.1784
Temps d'exécution : 12.930468082427979 secondes

1.3 Classifier KNN avec distance euclidienne

1.3.1 Classification binaire avec Bag-of-Words (Binaire) avec KNN - Version 1

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 54.78660 secondes
Accuracy (test) : 0.5559

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.4375
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625

- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625

Taux de bonne classification par fold : [np.float64(0.5625), np.float64(0.4375), np.float64(0.5625), np.float64(0.5625), np.float64(0.5625)]
Taux moyen de bonne classification : 0.5375
Écart-type : 0.0500

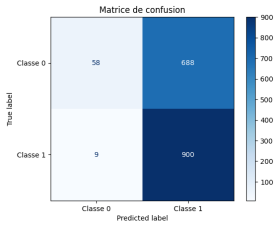

Figure 21: Confusion Matrix of KNN Version 1

1.3.2 Classification binaire avec Bag-of-Words (Comptage) avec KNN - Version 2

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 54.42367 secondes
Accuracy (test) : 0.5976

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.6250
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.4375
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.6875

Taux de bonne classification par fold : [np.float64(0.625), np.float64(0.4375), np.float64(0.5625), np.float64(0.5625), np.float64(0.6875)]
Taux moyen de bonne classification : 0.5750
Écart-type : 0.0829

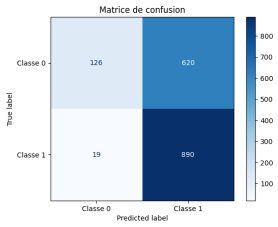

Figure 22: Confusion Matrix of KNN Version 2

1.3.3 Classification binaire avec Bag-of-Words (Fréquence) avec KNN - Version 3

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 53.55328 secondes
Accuracy (test) : 0.7535

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.4375
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.7500
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.6250
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5000
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.6250

Taux de bonne classification par fold : [np.float64(0.4375), np.float64(0.75), np.float64(0.625), np.float64(0.5), np.float64(0.625)]
Taux moyen de bonne classification : 0.5875
Écart-type : 0.1090

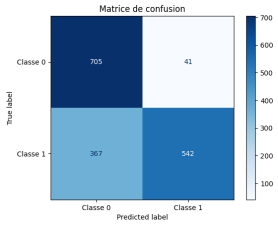

Figure 23: Confusion Matrix of KNN Version 3

1.3.4 Classification binaire avec Bag-of-Words (TF-IDF) avec KNN - Version 4

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 55.58179 secondes
Accuracy (test) : 0.7009

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.4375
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.5625
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.4375
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.6875

Taux de bonne classification par fold : [np.float64(0.4375), np.float64(0.5625), np.float64(0.5625), np.float64(0.4375), np.float64(0.6875)]
Taux moyen de bonne classification : 0.5375
Écart-type : 0.0935

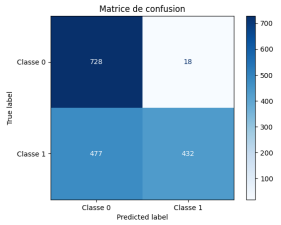

Figure 24: Confusion Matrix of KNN Version 4

1.3.5 Classification multi-classe avec KNN avec distance euclidienne (approche One-vs-All)

Entraînement de plusieurs classifieurs effectué en 0.00296 secondes
Prédiction sur test effectuée en 34.81226 secondes
Accuracy (KNN multi-classe avec PCA) : 0.1510

1.4 Classifier KNN avec distance cosinus

1.4.1 Classification binaire avec Bag-of-Words (Binaire) avec KNN - Version 1

Apprentissage effectué en 0.00018 secondes
Prédiction sur test effectuée en 37.24320 secondes
Accuracy (test) : 0.8532

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.7500
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375

Taux de bonne classification par fold : [np.float64(0.875), np.float64(0.875), np.float64(0.9375), np.float64(0.75), np.float64(0.9375)]
Taux moyen de bonne classification : 0.8750
Écart-type : 0.0685

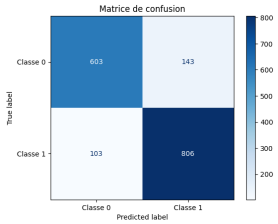

Figure 25: Confusion Matrix of KNN Version 1

1.4.2 Classification binaire avec Bag-of-Words (Comptage) avec KNN - Version 2

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 37.77043 secondes
Accuracy (test) : 0.8568

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8125
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(0.9375), np.float64(0.875), np.float64(0.8125), np.float64(0.875), np.float64(1.0)]
Taux moyen de bonne classification : 0.9000
Écart-type : 0.0637

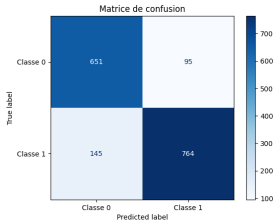

Figure 26: Confusion Matrix of KNN Version 2

1.4.3 Classification binaire avec Bag-of-Words (Fréquence) avec KNN - Version 3

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 38.13967 secondes
Accuracy (test) : 0.8568

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8125

- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(0.9375), np.float64(0.875), np.float64(0.8125), np.float64(0.875), np.float64(1.0)]
Taux moyen de bonne classification : 0.9000
Écart-type : 0.0637

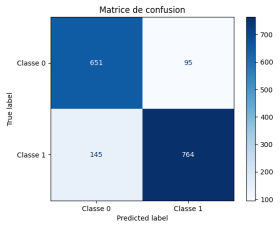

Figure 27: Confusion Matrix of KNN Version 3

1.4.4 Classification binaire avec Bag-of-Words (TF-IDF) avec KNN - Version 4

Apprentissage effectué en 0.00005 secondes
Prédiction sur test effectuée en 37.45100 secondes
Accuracy (test) : 0.8918

Validation croisée

- Itération 0: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.9375
- Itération 1: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.7500
- Itération 2: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 3: taille base app.= 70, taille base test=16, Taux de bonne classif: 0.8750
- Itération 4: taille base app.= 70, taille base test=16, Taux de bonne classif: 1.0000

Taux de bonne classification par fold : [np.float64(0.9375), np.float64(0.75), np.float64(0.875), np.float64(0.875), np.float64(1.0)]
Taux moyen de bonne classification : 0.8875

Écart-type : 0.0829

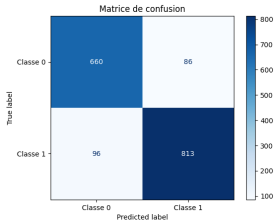

Figure 28: Confusion Matrix of KNN Version 4

1.4.5 Classification multi-classe avec KNN et distance cosinus (approche One-vs-All)

Entraînement de plusieurs classifieurs effectué en 72.93996 secondes
Prédiction sur test effectuée en 42.90721 secondes
Accuracy (KNN multi-classe cosinus avec PCA) : 0.1963

1.5 Classifier Naive Bayes

Accuracy (Naive Bayes multinomial): 0.2887

1.6 Classifier Arbre de décision

Figure 29: Decision Tree

2 Apprentissage non supervisé

2.1 Clustering hiérarchique

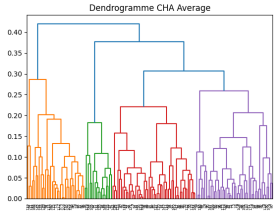

Figure 30: Average Linkage

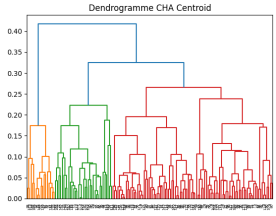

Figure 31: Centroid Linkage

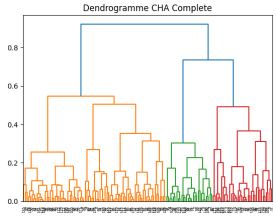

Figure 32: Complete Linkage

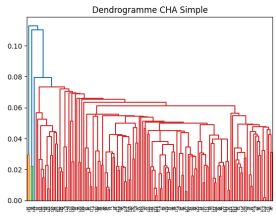

Figure 33: Single Linkage

- Le clustering hiérarchique avec la méthode "Average" (figure 30) montre une structure de clustering modérée, où les clusters sont formés en considérant la distance moyenne entre les points.
- Le clustering hiérarchique avec la méthode "Centroid" (figure 31) offre une autre perspective sur la structure des données, en utilisant le centroïde des clusters pour déterminer la distance.
- Le clustering hiérarchique avec la méthode "Complete" (figure 32) montre une approche plus stricte pour la formation des clusters, en utilisant la distance maximale entre les points des clusters.
- Le clustering hiérarchique avec la méthode "Single" (figure 33) tend à former des clusters plus petits et plus nombreux, en utilisant la distance minimale entre les points des clusters.

2.2 k-moyennes

2.2.1 Résultats avec k = 5

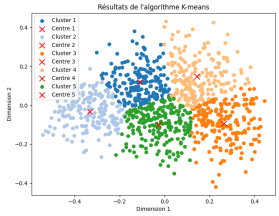

Figure 34: Clustering par k-moyennes avec k = 5

2.2.2 Résultats avec k = 10

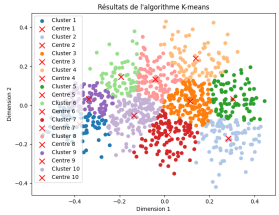

Figure 35: Clustering par k-moyennes avec k = 10

2.2.3 Résultats avec k = 20

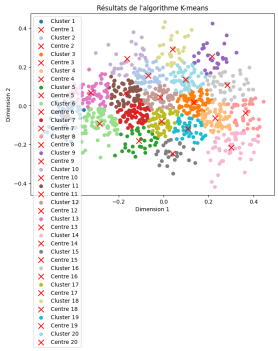

Figure 36: Clustering par k-moyennes avec k = 20

Métriques d'évaluation

- Score de silhouette : 0.3248
- Xie-Beni Index (manuel) : 0.4941
- Dunn Index (manuel) : 0.0193
- Indice de Rand Ajusté (ARI) : 0.0008
- Information Mutuelle Normalisée (NMI) : 0.0053

3 Résultats des métriques d'évaluation

1. **Score de silhouette** :
   - **Valeur** : 0.3248
   - **Interprétation** : Le score de silhouette mesure la similarité d'un échantillon avec son propre cluster par rapport aux autres clusters. Il varie entre -1 et 1, où une valeur plus élevée indique un meilleur clustering.
   - **Analyse** : Un score de 0.3248 indique un clustering modéré, avec une marge d'amélioration. Idéalement, on souhaite que ce score soit proche de 1.
2. **Indice de Xie-Beni** :
   - **Valeur** : 0.4941
   - **Interprétation** : L'indice de Xie-Beni évalue la compacité et la séparation des clusters. Une valeur plus faible indique un meilleur clustering.

- **Analyse** : Une valeur de 0.4941 suggère un clustering acceptable, mais avec une marge d'amélioration. Idéalement, on souhaite que cette valeur soit aussi faible que possible.

3. **Indice de Dunn** :
   - **Valeur** : 0.0193
   - **Interprétation** : L'indice de Dunn mesure la compacité des clusters et la séparation entre eux. Une valeur plus élevée indique un meilleur clustering.
   - **Analyse** : Une valeur de 0.0193 est très faible, indiquant un clustering peu performant. Idéalement, on souhaite que cette valeur soit aussi élevée que possible.
4. **Indice de Rand ajusté (ARI)** :
   - **Valeur** : 0.0008
   - **Interprétation** : L'ARI mesure la similarité entre le clustering obtenu et les vraies étiquettes. Il varie entre -1 et 1, où une valeur plus élevée indique un meilleur clustering.
   - **Analyse** : Une valeur de 0.0008, proche de 0, indique une similarité presque nulle entre le clustering obtenu et les vraies étiquettes. Idéalement, on souhaite que cette valeur soit proche de 1.
5. **Information Mutuelle Normalisée (NMI)** :
   - **Valeur** : 0.0053
   - **Interprétation** : Le NMI mesure la quantité d'information partagée entre le clustering obtenu et les vraies étiquettes. Il varie entre 0 et 1, où une valeur plus élevée indique un meilleur clustering.
   - **Analyse** : Une valeur de 0.0053 est très faible, indiquant une quantité presque nulle d'information partagée entre le clustering obtenu et les vraies étiquettes. Idéalement, on souhaite que cette valeur soit proche de 1.

4 Conclusion

Dans ce projet, nous avons comparé plusieurs variantes du Perceptron et du k-NN sur des tâches de classification binaire et multi-classe, avec différentes représentations de textes (binaire, comptage, fréquence, TF-IDF).

L'ajout d'un biais dans le Perceptron améliore clairement les performances, en particulier avec les représentations binaire et TF-IDF. En revanche, les résultats sont moins bons en multi-classe, ce qui montre les limites du Perceptron "one-vs-all". Le k-NN donne de bons résultats avec k=3, surtout avec la distance cosinus et les représentations adaptées comme TF-IDF. Ce classifieur reste simple mais coûteux à grande échelle.

Le modèle Naive Bayes, bien que simple, a montré une précision de 0.2887, ce qui indique qu'il peut être moins performant que d'autres modèles pour ce type de données. L'arbre de décision, illustré dans la figure 29, offre une approche interprétable, mais son efficacité dépend fortement de la qualité des caractéristiques et de la profondeur de l'arbre.

La validation croisée confirme la stabilité des modèles, même si la classification multi-classe reste plus difficile. Nos résultats montrent l'importance du choix de la représentation et du classifieur.

Enfin, il serait intéressant d'explorer l'impact de l'augmentation de la taille des données ou de l'utilisation de techniques de sélection de caractéristiques pour améliorer les performances des modèles. De plus, l'expérimentation avec d'autres algorithmes ou l'optimisation des hyperparamètres pourrait également conduire à des améliorations significatives.