Projet SDD

Yuxiang ZHANG et Antoine LECOMTE
Groupe 2

Mai 2025

1 Apprentissage supervisé

1.1 Description du Dataset

Le dataset utilisé dans ce projet contient initialement 18446 lignes et 2 colonnes. Un prétraitement de nettoyage a été appliqué pour éliminer une liste de mots inutiles (stopwords) contenant 198 mots. Après ce nettoyage, il restait 90850 mots. Cependant, en raison de temps d'exécution trop longs pour certaines méthodes, un nouveau raccourcissement du dataset a été effectué en appliquant un filtre conservant 11928 mots pour 18466 messages. Ce filtre a retenu les mots anglais, les mots de 3 à 20 lettres, et ceux apparaissant au moins 5 fois dans le dataset. Les analyses présentées dans ce Poster ont été effectuées sur ce sous-ensemble de données.

1.2 Analyse des Performances de Classification

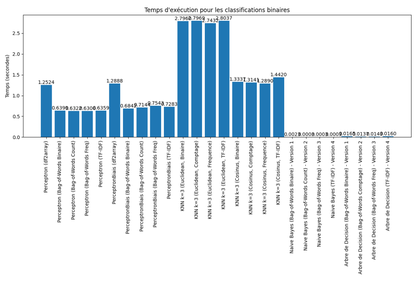

Figure 1: Temps d'exécution pour les classifications binaires

Remarques :

- Le classifieur Naive Bayes est le plus rapide avec un temps d'exécution quasi-nul.
- L'arbre de décision est presque aussi rapide à l'exécution, mais la différence reste notable.
- Les classifieurs Perceptron et PerceptronBiais ont des temps d'exécution modérés.
- Les classifieurs KNN sont les plus lents, en particulier ceux utilisant la distance euclidienne.

Conclusion :

- Meilleur classifieur : Naive Bayes
- Pire classifieur : KNN Euclidien (TF-IDF)

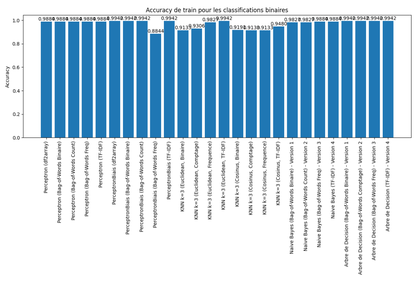

Figure 2: Accuracy de train pour les classifications binaires

Remarques :

- La plupart des classifieurs ont une accuracy de train très élevée, proche de 1.
- La représentation TF-IDF semble être, pour l'ensemble des classifieurs, la meilleure.

Conclusion :

- Meilleur classifieur : Perceptron et Arbre de décision
- Pire classifieur : PerceptronBiais (Freq)

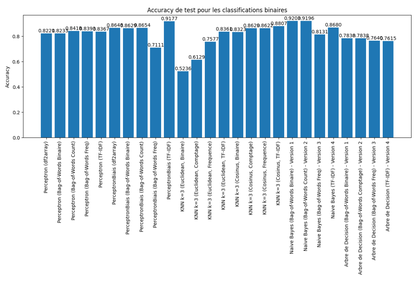

Figure 3: Accuracy de test pour les classifications binaires

Remarques :

- Les classifieurs KNN cosinus et perceptron sont stables selon la méthode de représentation.
- Les classifieurs KNN avec distance euclidienne ont des performances de test plus faibles et qui varient davantage par rapport à la méthode de représentation que pour les autres classifieurs.

Conclusion :

- Meilleur classifieur : Naive Bayes (Binaire)
- Pire classifieur : KNN Euclidien, Binaire)

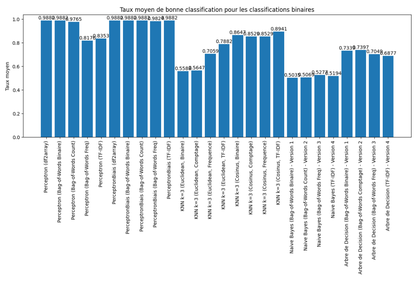

Figure 4: Taux moyen de bonne classification pour les classifications binaires

Remarques :

- Les classifieurs PerceptronBiais, Perceptron et KNN distance cosinus ont des taux moyens de bonne classification élevés.
- Les classifieurs KNN avec distance euclidienne, les arbres de décision et Naive Bayes ont des taux moyens plus faibles.

Conclusion :

- Meilleur classifieur : PerceptronBiais. Il est excellent et clairement le meilleur pour toutes les représentations.
- Pire classifieur : Naive Bayes. Très en dessous des autres classifieurs.

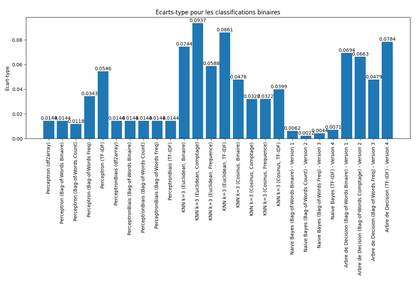

Figure 5: Écarts-type pour les classifications binaires

Remarques :

- Un écart-type élevé indique une variance entre la moyenne des points plus importante, pour un classifieur efficace et stable, il vaut mieux qu'il soit minimal quelle que soit la représentation choisie.
- Les classifieurs Naive Bayes ont des écarts-types parmi les plus faibles mais ils varient selon la représentation.
- Les classifieurs KNN avec distance euclidienne ont des écarts-types plus élevés, variant également beaucoup aussi avec la méthode de représentation.

Conclusion :

- Meilleur classifieur : PerceptronBiais. écart-type faible et stable avec changement de représentation des données.
- Pire classifieur : KNN Euclidien. Valeurs élevées et trop changeantes.

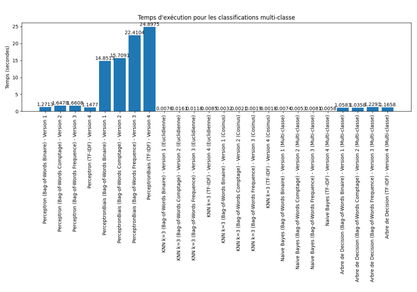

Figure 6: Temps d'exécution pour les classifications multi-classe

Remarques :

- Les classifieurs Perceptron et les arbres de décision ont des temps d'exécution modérés.
- Les classifieurs PerceptronBiais sont significativement plus lents.
- Les autres ont un temps quasi-nul.

Conclusion :

- Meilleur classifieur : KNN (Euclidien et Cosinus) et Naive Bayes.
- Pire classifieur : PerceptronBiais. Très lent, surtout avec TF-IDF.

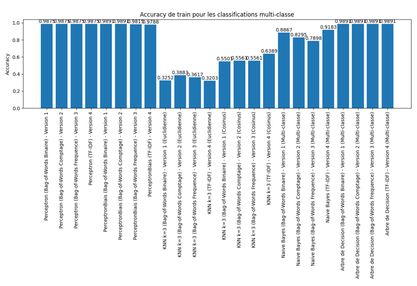

Figure 7: Accuracy de train pour les classifications multi-classe

Remarques :

- Tous les classifieurs hormis les KNN et Naive Bayes ont une accuracy de train très élevée, proche de 1.

Conclusion :

- Meilleur classifieur : Arbre de décision (mais très proche du perceptron et perceptron biais).
- Pire classifieur : KNN Euclidien.

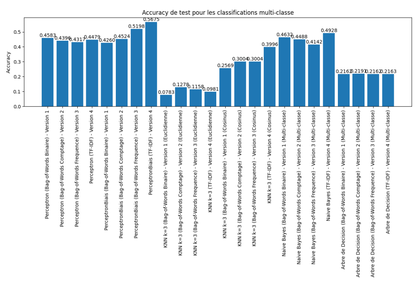

Figure 8: Accuracy de test pour les classifications multi-classe

Remarques :

- Les classifieurs PerceptronBiais ont des performances de test légèrement meilleures que le perceptron.
- Les classifieurs KNN avec distance euclidienne et arbres de décision ont des performances de test plus faibles.

Conclusion :

- Meilleur classifieur : PerceptronBiais (surtout avec TF-IDF) mais talonné par le perceptron et Naive Bayes.
- Pire classifieur : KNN Euclidien

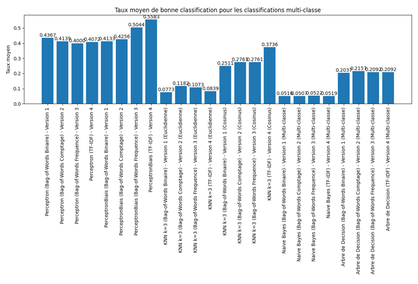

Figure 9: Taux moyen de bonne classification pour les classifications multi-classe

Remarques :

- Les classifieurs PerceptronBiais ont des taux moyens de bonne classification légèrement meilleurs que les perceptrons, et KNN avec distance cosinus est un peu derrière et suivi de près par les arbres de décision.

Conclusion :

- Meilleur classifieur : PerceptronBiais (avec TF-IDF au mieux).
- Pire classifieur : Naive Bayes (Bag-of-Words count pour le pire, bien que les trois autres représentations soient également mauvaises).

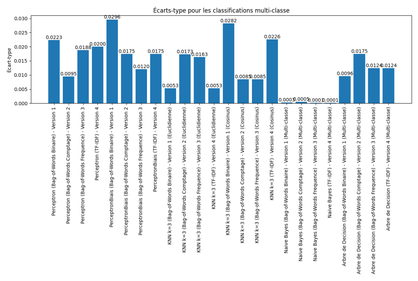

Figure 10: Écarts-type pour les classifications multi-classe

Remarques :

- Pas de réelle démarcation pour un classifieur plutôt qu'un autre.
- Le classifieur Perceptron biais en représentation binaire atteint le plus grand écart sur ses données.

Conclusion :

- Meilleur classifieur : Naive Bayes. Très faible écart entre les représentations de données et valeurs très faibles.
- Pire classifieur : Tous les autres. Les écarts-types varient beaucoup avec la représentation de données utilisée, car pour rappel pour avoir un classifieur efficace et stable, il vaut mieux qu'il donne un écart-type minimal quelle que soit la représentation choisie.

1.2.1 Conclusion

Le critère le plus important étant le résultat final, soit le taux de bonne classification, le meilleur classifieur pour cela est donc le perceptron biais, en classification binaire tout comme en multi-classe. Les matrices de confusion, visibles dans l'ensemble du code python associé, nous aident à se représenter les taux de bonnes et mauvaises classifications effectuées dans chaque classe, pour les classifications binaire et multi-classe.

1.3 Analyse des Performances de Classification de KNN selon différentes valeurs de k

Nous avons évalué les performances de classification en utilisant l'algorithme KNN avec différentes valeurs de k (3, 5, 135) et pour l'ensemble des méthodes de représentation des données utilisées précédemment. Les résultats obtenus pour les classifications binaires et multi-classes, en utilisant les distances euclidienne et cosinus, sont présentés ci-dessous.

1.3.1 Classification Binaire avec Distance Euclidienne

Le meilleur k pour la classification binaire avec distance euclidienne est **TF-IDF avec** $k = 3$, qui donne la meilleure performance avec un taux moyen de bonne classification de 0.7882.

1.3.2 Classification Multi-classe avec Distance Euclidienne

Le meilleur k pour la classification multi-classe avec distance euclidienne est **Comptage avec** $k = 3$ *ou* $k = 5$, qui donne la meilleure performance avec un taux moyen de 0.1182.

1.3.3 Classification Binaire avec Distance Cosinus

Le meilleur k pour la classification binaire avec distance cosinus est **TF-IDF avec** $k = 5$, qui donne la meilleure performance avec un taux moyen de bonne classification de 0.9059.

1.3.4 Classification Multi-classe avec Distance Cosinus

Le meilleur k pour la classification multi-classe avec distance cosinus est **TF-IDF avec** $k = 135$, qui donne la meilleure performance avec un taux moyen de 0.4760.

1.4 Conclusion

Dans les perspectives d'améliorations pour notre projet, nous pouvons chercher une meilleure valeur de k pour une meilleure classification. Ces résultats nous guideraient dans l'optimisation de nos modèles de classification pour de meilleures performances, mais en l'absence d'autres résultats, en mettant en avant la classification multi-classe (qui est souvent la plus utile), il vaut mieux utiliser $k = 135$ car nous avons vu que le classifieur KNN avec distance cosinus est plus performant que celui qui utilise la distance euclidienne.

2 Apprentissage non supervisé

2.1 Clustering hiérarchique

Nous avons récupéré un échantillon des 150 premiers messages du dataset pour afficher les clustering hiérarchiques de façon à ce que les dendrogrammes puissent être lisibles.

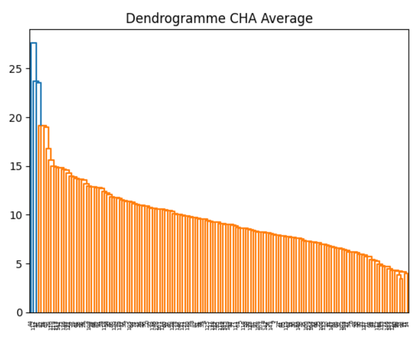

Figure 11: Average Linkage

Le clustering hiérarchique avec la méthode "Average" montre une structure de clustering modérée, où les clusters sont formés en considérant la distance moyenne entre les points.

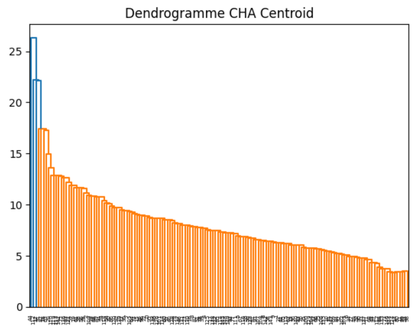

Figure 12: Centroid Linkage

Le clustering hiérarchique avec la méthode "Centroid" offre une autre perspective sur la structure des données, en utilisant le centroïde des clusters pour déterminer la distance.

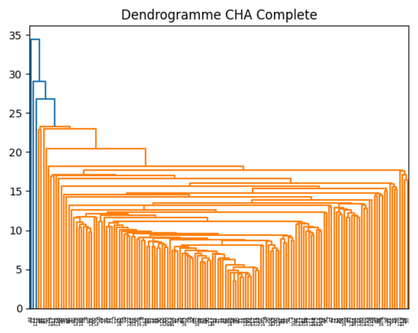

Figure 13: Complete Linkage

Le clustering hiérarchique avec la méthode "Complete" montre une approche plus stricte pour la formation des clusters, en utilisant la distance maximale entre les points des clusters.

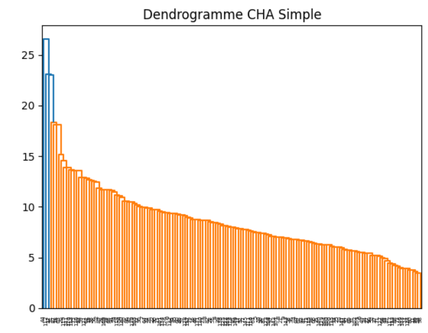

Figure 14: Single Linkage

Le clustering hiérarchique avec la méthode "Single" tend à former des clusters plus petits et plus nombreux, en utilisant la distance minimale entre les points des clusters.

2.1.1 Conclusion

Quelle que soit l'approche utilisée pour le Clustering, on obtient un unique cluster à partir d'une structure arborescente de clusters où chaque point de données commence dans son propre cluster, et les clusters sont successivement fusionnés jusqu'à ce qu'il ne reste qu'un seul cluster contenant toutes les données.

2.2 k-moyennes

Pour le clustering par k-moyennes, nous avons pris l'ensemble des messages et nous avons appliqué une réduction de dimensionnalité avec PCA pour réduire les données à deux dimensions afin de nous permettre de visualiser des ensembles de données multidimensionnels. Cela nous permet d'identifier des motifs, des clusters ou des anomalies dans les données. Nous avons ensuite exécuté l'algorithme k-moyennes avec différents nombres de clusters (k = 5, 10, 20).

2.2.1 Résultats avec k = 5

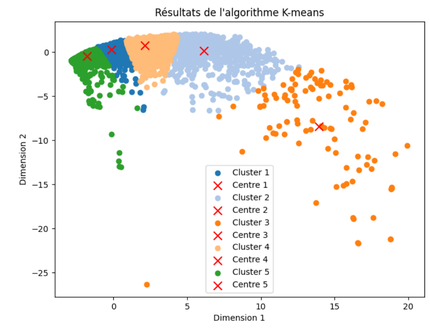

Figure 15: Clustering par k-moyennes avec k = 5

Métriques d'évaluation

- Score de silhouette : 0.4846
- Xie-Beni Index (manuel) : 0.3101
- Dunn Index (manuel) : 0.0001
- Indice de Rand Ajusté (ARI) : 0.0064
- Information Mutuelle Normalisée (NMI) : 0.0242

2.2.2 Résultats avec k = 10

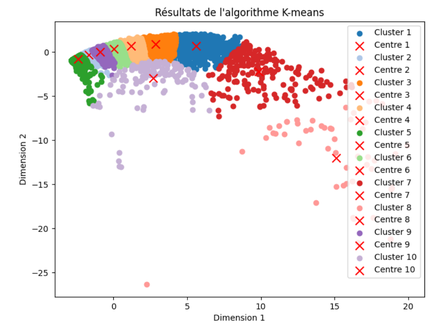

Figure 16: Clustering par k-moyennes avec k = 10

Métriques d'évaluation

- Score de silhouette : 0.4010
- Xie-Beni Index (manuel) : 0.7242
- Dunn Index (manuel) : 0.0000
- Indice de Rand Ajusté (ARI) : 0.0065
- Information Mutuelle Normalisée (NMI) : 0.0272

2.2.3 Résultats avec k = 20

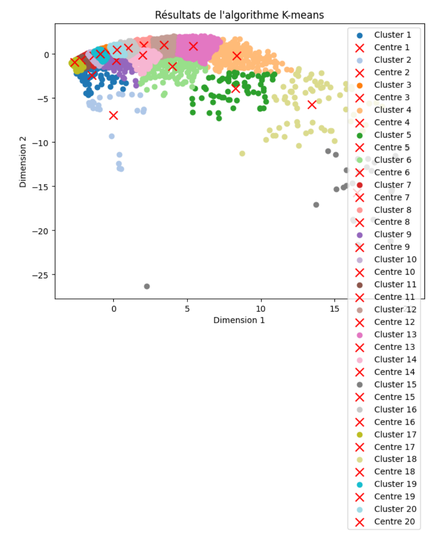

Figure 17: Clustering par k-moyennes avec k = 20

Métriques d'évaluation

- Score de silhouette : 0.3523
- Xie-Beni Index (manuel) : 1.2682
- Dunn Index (manuel) : 0.0001
- Indice de Rand Ajusté (ARI) : 0.0063
- Information Mutuelle Normalisée (NMI) : 0.0342

2.3 Résultats des métriques d'évaluation

1. **Score de silhouette** :
   - **Valeur** : 0.4846 (k=5), 0.4010 (k=10), 0.3523 (k=20)
   - **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. Un score de 0.4846 pour k=5 indique un clustering modéré, avec une marge d'amélioration. Idéalement, on souhaite que ce score soit proche de 1, ce qui n'est pas le cas.
2. **Indice de Xie-Beni** :
   - **Valeur** : 0.3101 (k=5), 0.7242 (k=10), 1.2682 (k=20)
   - **Interprétation** : L'indice de Xie-Beni évalue la compacité et la séparation des clusters. Une valeur plus faible indique un meilleur clustering. Une valeur de 0.3101 pour k=5 est un clustering acceptable, mais avec une bonne marge d'amélioration. Idéalement, on souhaite que cette valeur soit aussi faible que possible.
3. **Indice de Dunn** :
   - **Valeur** : 0.0001 (k=5), 0.0000 (k=10), 0.0001 (k=20)
   - **Interprétation** : L'indice de Dunn mesure la compacité des clusters et la séparation entre eux. Une valeur plus élevée indique un meilleur clustering. Une valeur de 0.0001 pour k=5 est très faible, indiquant un clustering très peu performant. Idéalement, on souhaite que cette valeur soit aussi élevée que possible.
4. **Indice de Rand ajusté (ARI)** :
   - **Valeur** : 0.0064 (k=5), 0.0065 (k=10), 0.0063 (k=20)
   - **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. Une valeur de 0.0064 pour k=5, 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** : 0.0242 (k=5), 0.0272 (k=10), 0.0342 (k=20)
   - **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. Une valeur de 0.0242 pour k=5 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.

En résumé, les résultats des métriques d'évaluation montrent que le clustering actuel n'est pas optimal. Des améliorations sont nécessaires, telles que l'ajustement des paramètres comme le seuil de convergence (epsilon) défini à 1/10000 et il serait intéressant d'essayer d'autres valeurs de k pour essayer de trouver de meilleurs résultats. Aussi, le problème peut simplement venir du dataset en question, les messages contiennent des mots trop répétitifs, difficilement classifiables.

3 Conclusion générale

Dans ce projet, nous avons exploré et évalué plusieurs méthodes de classification supervisée et non supervisée sur un dataset de messages textuels. Ce projet nous a permis d'acquérir une expérience pratique dans l'application de méthodes de classification et de clustering sur des données textuelles. Les résultats obtenus ouvrent la voie à des améliorations futures pour optimiser les performances des modèles et obtenir des résultats plus précis et fiables.