Rapport TP3

Techniques avancées en Intelligence Artificielle

IFT-7025

Présenté par:

David Poisson (905 302 625)

Julien Lafrance (111 268 508)

Équipe #31

Définitions

Cette section décrit les termes qui seront utiliser fréquemment lors de ce travail. Il est donc vital qu'une définition claire soit établie afin d'éviter toute confusion.

Exactitude (Accurary): Le pourcentage des exemples bien classifiés par rapport au nombre total des exemples présentés à l'algorithme.

La précision (Precision): Le pourcentage des exemples bien classifiés dans une classe quelconque par rapport au nombre total des individus classifiés dans cette même classe par l'algorithme.

Rappel (Recall): Le pourcentage des exemples bien classifiés dans une classe quelconque par rapport au nombre total des individus qui appartiennent réellement à cette même classe.

F1-score: Le F1-score est une mesure qui met en relation la précision et le rappel pour permettre d'avoir une évaluation bien balancée. Le F1-score pondère de façon égale la précision et le rappel. Ces deux mesures ont donc la même importance sur la valeur du F1-Score

Matrice de confusion (Confusion matrix): La matrice de confusion est une classification de la performance d'un classificateur sur des données d'essais après avoir après sur des données d'entraînement pour une classe donnée. 2 éléments (actuel et prédit) ont chacun 2 valeurs possibles: Donnant une matrice 2x2. Les 4 cases que forment cette matrice s'appelle:

- TruePositive: Prédit correctement la classe réelle
- FalsePositive: Prédit incorrectement la classe réelle
- TrueNegative: Prédit la présence de la classe réelle alors qu'elle n'est pas présente
- FalseNegative: Prédit l'absence de la classe réelle alors qu'elle est présente

Il est possible de calculer l'ensemble des métriques mentionnés ci-dessus avec les formules suivantes pour chacune des classes possibles:

Exactitude = TruePositive + TrueNegative / Nombre Total d'élément

Précision = TruePositive / (TruePositive + FalsePositive)

Rappel = TruePositive / (TruePositive + FalseNegative)

F1-score = 2 * ((précision * rappel) / (précision + rappel))

K plus proches voisins (K-nearest neighbors)

Mesure de distance

Méthode choisie: Distance euclidienne (Minkowski)

Cette mesure de distance a été choisie, car elle permet d'augmenter la dissimilarité entre les observations ayant de grandes différences dans les valeurs d'une même caractéristique.

L'augmentation du temps de calcul en ajoutant la mise au carré des différences intra-caractéristique a un impact minime sur le temps de prédiction.

Pseudo-code de l'algorithme

Entrainement

Stockage des points.

Prédiction

On itère sur chaque exemple d'entraînement

Calcule la distance avec le point à prédire

On stocke la distance calculée ainsi que la classe dans une liste de tuple

On trouve les k distances les plus courtes avec le nouveau point.

Parmi les k distances les plus courtes, on calcule la classe ayant le plus d'observations. Cette classe est la classe prédite pour le nouveau point.

Évaluation

Évaluations de la méthode: Voir annexe KNN. Les métriques ainsi que la matrice de confusion sera affichée lors de l'exécution.

On voit (voir annexe) que l'algorithme des K plus proches voisins fonctionne bien avec un nombre faible de caractéristiques. Lorsque le jeu de données augmente en nombre de dimensions, moins la prédiction est exacte. Cette perte de précision semble être causé par l'augmentation du nombre de dimension ce qui augmente la possibilité d'obtenir une grande distance entre les points.

On ne voit pas de débalancement entre les prédictions pour les différentes classes. Toutes les classes sont choisies relativement également.

Cross-validation (IFT-7025)

Nous avons implémenté la cross-validation en utilisant la meilleure moyenne des scores F1 afin de choisir la meilleure valeur de K.

Les meilleures valeurs de k trouvées sont:

Iris: 5, Wine: 5, Abalone: 5

Comparatif avec scikit-learn

Comparaison entre les méthodes codées et scikit-learn : Voir annexe KNN

Les métriques d'évaluation sont identiques.

Cependant, le temps de prédiction du modèle est à 2 ordres de grandeur pour des jeux de données de taille réduite. Comme première implémentation, cette différence marquée est normale. Plusieurs optimisations comme le retrait du triage pour trouver les points les plus proches et l'utilisation d'une heuristique pour réduire le nombre de points dont on calcule la distance réduiraient drastiquement le temps de prédiction.

Classification Naïf Bayésien

Pseudo-code de l'algorithme d'entraînement

Initialise les étiquettes d'entraînement

On itère sur chaque classe possible et calcule:

- La probabilité de la classe
- La moyenne ainsi que la variance pour chaque caractéristique

Pour chaque classe, on identifie les cas d'entraînement appartenant à cette classe

> Pour chaque caractéristique, on extrait les valeurs appartenant à la classe et on calcule et sauvegarde:
>
> - La moyenne
> - La variance

Pseudo-code de l'algorithme de prédiction

Itère pour chaque classe possible

Pour chaque classe, on obtient la probabilité de cette classe

Pour chaque caractéristique, on calcule la probabilité conditionnelle en utilisant une distribution gaussienne avec la moyenne et la variance de celle-ci

Si la probabilité totale calculée pour la classe actuelle est supérieure à la probabilité maximale, on met à jour la probabilité maximale ainsi que la meilleure classe prédite

Une fois l'itération terminée, on retourne la meilleure classe prédite

Évaluation

Comparaison entre les méthodes codées et scikit-learn : Voir annexe Bayes Naïf. Les métriques ainsi que la matrice de confusion sera affichée lors de l'exécution.

On peut constater que les métriques sont presque d'identique. Cependant, le classificateur implémenté par scikit-learn est clairement plus rapide.

Discussion des résultats

Lorsque les données possédaient plus de 2 classes, nous avons eu de la difficulté lors du calcul des métriques. Notamment au niveau de la matrice de confusion, car nous devions tenir compte de chaque classe.

Certaines données ont aussi dû être modifiées telle que la classe prédite du dataset Iris par exemple. Nous avons effectué une substitution de chaîne de caractères pour un entier afin de faciliter le traitement. De plus, la donnée du Sexe a dû être modifié afin de permettre un traitement par les classificateurs.

Nous avons remarqué que la taille du jeu de données influence grandement sur la performance. De plus, la différence en termes de temps d'exécution est marquée entre Knn et BayesNaif. Le classificateur Bayes s'exécute très rapidement, même pour les datasets plus volumineux, alors que Knn prend beaucoup de temps à s'exécuter. Et ce, même pour un dataset de taille raisonnable, tel que le dataset Wine.

Accuracy

Accuracy (moyenne) sur data de test

	Bayes Naif	**Knn**
Iris	1.000	0.967
Wine	0.813	0.787
Abalone	0.581	0.840

La différence la plus grande se situe au niveau du dataset **Abalone**. On voit que notre classificateur Knn performe nettement mieux bien que le classificateur Bayes Naïf (un accuracy 50% plus élevé). Sinon, pour les dataset **Iris** et **Wine**, l'accuracy est très similaire entre les deux classificateurs.

Precision

Précision sur data de test (pour chaque classe)

	Bayes Naïf			**Knn**		
Iris	1.000	1.000	1.000	1.000	1.000	0.910
Wine	0.889	0.719		0.828	0.721	
Abalone	0.426	0.871	0.212	0.646	0.876	0.575

Pour les datasets **Iris** et **Wine**, les métriques sont très similaires entre le classificateur Bayes Naïf et Knn, peu importe la classe. Pour le dataset **Abalone**, on constate que Knn est meilleur pour l'ensemble des classes. Pour la classe #2, on pourrait conclure que la différence de 0.005 est négligeable, mais pour les autres classes, la différence est beaucoup plus marquée.

Recall

Recall sur data de test (pour chaque classe)

	Bayes Naïf			**Knn**		
Iris	1.000	1.000	1.000	1.000	0.889	1.000
Wine	0.786	0.841		0.826	0.725	
Abalone	0.954	0.543	0.505	0.593	0.933	0.299

Pour le dataset **Iris**, les résultats sont très similaires. On note une petite baisse du rappel entre le Knn par rapport au Bayes Naïf. Pour ce qui est du dataset **Wine**, il y a une différence par rapport à chacune des deux classes. Finalement, pour **Abalone**, on constate que Bayes Naïf est supérieur pour 2 classes sur 3 comparativement à Knn.

F1 score

F1 score sur data de test (pour chaque classe)

	Bayes Naïf			**Knn**		
Iris	1.000	1.000	1.000	1.000	0.941	0.952
Wine	0.840	0.775		0.827	0.723	
Abalone	0.589	0.669	0.305	0.618	0.904	0.393

Le score du classificateur Bayes Naïf pour le dataset **Iris** est superbe, alors que le Knn un peu moins bon. Pour ce qui est du dataset **Wine**, les résultats sont très semblables sans claire vainqueur. Finalement, pour **Abalone**, le Knn a constamment de meilleur résultat.

Apprentissage suite au développement de notre solution

Lorsque nous avons comparé nos implémentations avec scikit-learn, nous avons constaté des différences avec le découpage des fonctions dans le code de scikit-learn. La principale différence est la combinaison de la méthode de prédiction et d'évaluation. L'architecture développée a été formée autour des attentes du cours. Celle-ci est donc peu extensible et rigide.

Il aurait été souhaitable d'éventuellement apporter les modifications suivantes dans notre code:

- Retourner un objet ConfusionMatrix lors de l'appel à la fonction evaluate() des classificateurs ou simplement d'être capable de créer un tel objet avec l'ensemble des prédictions ainsi que l'ensemble des résultats réels.
- Le fait d'utiliser la méthode predictArray(), appelée via la méthode d'évaluation porte à confusion, rendre la méthode predict() agnostique au format d'appel serait avantageux.
- On constate la différence de performance entre nos implémentations et celles de scikit-learn. Il serait intéressant de refaire une vérification dans le code afin d'améliorer les temps d'exécutions. Cependant, dans une situation de prototypage, les temps d'exécutions actuels sont suffisants.
- Le code du chargement des datasets pourrait être plus générique au lieu d'être dupliquer pour chaque dataset.

Conclusion

Lorsqu'il y a uniquement 2 classes ou que le jeu de données est limité en nombre, les deux classificateurs performent de manière similaire. Cependant, lorsqu'il y a un volume important de données ou qu'il y a plus de 2 classes, Knn donne de meilleurs résultats. Ceci vient cependant au détriment du temps d'exécution.

Annexes

Évaluations Modèles sur données de tests

Gauche: Notre implémentation et l'implémentation scikit-learn à droite.

KNN

Iris

```
Evaluating KNN classifier on test iris

Elapsed time: 0.014998912811279297

Evaluation metrics for class: 0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       11      |       0       | Positive
|---------------|---------------|
|       0       |       19      | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.9666666666666667
Precision: 1.0
Recall: 0.8888888888888888
F1-score: 0.9411764705882353

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       8       |       1       | Positive
|---------------|---------------|
|       0       |       21      | Negative
|---------------|---------------|

Evaluation metrics for class: 2
Accuracy: 0.9666666666666667
Precision: 0.9090909090909091
Recall: 1.0
F1-score: 0.9523809523809523

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       10      |       0       | Positive
|---------------|---------------|
|       1       |       19      | Negative
|---------------|---------------|

mean accuracy: 0.9666666666666667
elapsed_time: 0.014998912811279297
```

```
Evaluating Sci-kit KNN classifier on test iris

elapsed_time: 0.003001689910888672

Evaluation metrics for class: 0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       11      |       0       | Positive
|---------------|---------------|                 Actual
|       0       |       19      | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.9666666666666667
Precision: 1.0
Recall: 0.8888888888888888
F1-score: 0.9411764705882353

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       8       |       1       | Positive
|---------------|---------------|                 Actual
|       0       |       21      | Negative
|---------------|---------------|

Evaluation metrics for class: 2
Accuracy: 0.9666666666666667
Precision: 0.9090909090909091
Recall: 1.0
F1-score: 0.9523809523809523

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       10      |       0       | Positive
|---------------|---------------|                 Actual
|       1       |       19      | Negative
|---------------|---------------|

mean accuracy: 0.9666666666666667
```

Abalone

```
Evaluating KNN classifier on test abalone

Elapsed time prediction: 10.242213249206543

Evaluation metrics for class: 0
Accuracy: 0.9246411483253588
Precision: 0.6455696202531646
Recall: 0.5930232558139535
F1-score: 0.6181818181818183

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       51      |       35      | Positive
|---------------|---------------|
|       28      |       722     | Negative
|---------------|---------------|
Actual

Evaluation metrics for class: 1
Accuracy: 0.8397129186602871
Precision: 0.8758716875871687
Recall: 0.9331352154531947
F1-score: 0.9035971223021582

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       628     |       45      | Positive
|---------------|---------------|
|       89      |       74      | Negative
|---------------|---------------|
Actual

Evaluation metrics for class: 2
Accuracy: 0.9150717703349283
Precision: 0.575
Recall: 0.2987012987012987
F1-score: 0.39316239316239315

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       23      |       54      | Positive
|---------------|---------------|
|       17      |       742     | Negative
|---------------|---------------|
Actual

mean accuracy: 0.8397129186602871
```

```
Evaluating Sci-kit KNN classifier on test abalone

elapsed_time: 0.04100227355957031

Evaluation metrics for class: 0
Accuracy: 0.9246411483253588
Precision: 0.6455696202531646
Recall: 0.5930232558139535
F1-score: 0.6181818181818183

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       51      |       35      | Positive
|---------------|---------------|                Actual
|       28      |       722     | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.8397129186602871
Precision: 0.8758716875871687
Recall: 0.9331352154531947
F1-score: 0.9035971223021582

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       628     |       45      | Positive
|---------------|---------------|                Actual
|       89      |       74      | Negative
|---------------|---------------|

Evaluation metrics for class: 2
Accuracy: 0.9150717703349283
Precision: 0.575
Recall: 0.2987012987012987
F1-score: 0.39316239316239315

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       23      |       54      | Positive
|---------------|---------------|                Actual
|       17      |       742     | Negative
|---------------|---------------|

mean accuracy: 0.8397129186602871
```

Wine

```
Evaluating KNN classifier on test wine

Elapsed time: 4.134838342666626

Evaluation metrics for class: 0
Accuracy: 0.7870370370370371
Precision: 0.8283132530120482
Recall: 0.8258258258258259
F1-score: 0.8270676691729323

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       275     |       58      | Positive
|---------------|---------------|
|       57      |       150     | Negative
|---------------|---------------|

Actual

Evaluation metrics for class: 1
Accuracy: 0.7870370370370371
Precision: 0.7211538461538461
Recall: 0.7246376811594203
F1-score: 0.7228915662650603

Confusion matrix

            Predicted
    Positive    |    Negative
|---------------|---------------|
|       150     |       57      | Positive
|---------------|---------------|
|       58      |       275     | Negative
|---------------|---------------|

Actual

mean accuracy: 0.7870370370370371
```

```
Evaluating Sci-kit KNN classifier on test wine

elapsed_time: 0.017000913619995117

Evaluation metrics for class: 0
Accuracy: 0.7870370370370371
Precision: 0.8283132530120482
Recall: 0.8258258258258259
F1-score: 0.8270676691729323

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       275     |       58      | Positive
|---------------|---------------|
|       57      |       150     | Negative
|---------------|---------------|

Actual

Evaluation metrics for class: 1
Accuracy: 0.7870370370370371
Precision: 0.7211538461538461
Recall: 0.7246376811594203
F1-score: 0.7228915662650603

Confusion matrix

    Predicted
Positive        |    Negative
|---------------|---------------|
|       150     |       57      | Positive
|---------------|---------------|
|       58      |       275     | Negative
|---------------|---------------|

Actual

mean accuracy: 0.7870370370370371
```

Bayes Naif

Iris

```
Training NaifBayes classifier on train iris

Elapsed time: 0.0012352466583251953

Evaluating BayesNaif classifier on test iris

Elapsed time prediction: 0.0019998550415039062

Evaluation metrics for class: 0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix
           Predicted
    Positive     |    Negative
|----------------|----------------|
|       11       |       0        | Positive
|----------------|----------------|              Actual
|       0        |       19       | Negative
|----------------|----------------|

Evaluation metrics for class: 1
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix
           Predicted
    Positive     |    Negative
|----------------|----------------|
|       9        |       0        | Positive
|----------------|----------------|              Actual
|       0        |       21       | Negative
|----------------|----------------|

Evaluation metrics for class: 2
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix
           Predicted
    Positive     |    Negative
|----------------|----------------|
|       10       |       0        | Positive
|----------------|----------------|              Actual
|       0        |       20       | Negative
|----------------|----------------|

mean accuracy: 1.0
```

```
Training scikit GaussianNB classifier on train iris

Elapsed time training: 0.002010345458984375

Evaluating Sci-kit GaussianNB classifier on test iris

Elapsed Time predicting: 0.001001119613647461

Evaluation metrics for class: 0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix
    Predicted
Positive        |    Negative
|---------------|----------------|
|       11      |       0        | Positive
|---------------|----------------|              Actual
|       0       |       19       | Negative
|---------------|----------------|

Evaluation metrics for class: 1
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix
    Predicted
Positive        |    Negative
|---------------|----------------|
|       9       |       0        | Positive
|---------------|----------------|              Actual
|       0       |       21       | Negative
|---------------|----------------|

Evaluation metrics for class: 2
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0

Confusion matrix
    Predicted
Positive        |    Negative
|---------------|----------------|
|       10      |       0        | Positive
|---------------|----------------|              Actual
|       0       |       20       | Negative
|---------------|----------------|

mean accuracy: 1.0
```

Abalone

```
Training NaifBayes classifier on train abalone

Elapsed time training: 0.03901815414428711

Evaluating BayesNaif classifier on test abalone

Elapsed time prediction: 0.059020042419433594

Evaluation metrics for class: 0
Accuracy: 0.861244019138756
Precision: 0.4256410256410256
Recall: 0.9540229885057471
F1-score: 0.5886524822695035

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       83      |       4       | Positive
|---------------|---------------|              Actual
|       112     |       637     | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.5849282296650717
Precision: 0.8709677419354839
Recall: 0.5433436532507739
F1-score: 0.669208770257388

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       351     |       295     | Positive
|---------------|---------------|              Actual
|       52      |       138     | Negative
|---------------|---------------|

Evaluation metrics for class: 2
Accuracy: 0.7165071770334929
Precision: 0.2184873949579832
Recall: 0.5048543689320388
F1-score: 0.30498533724340177

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       52      |       51      | Positive
|---------------|---------------|              Actual
|       186     |       547     | Negative
|---------------|---------------|

mean accuracy: 0.5813397129186603
```

```
Training scikit GaussianNB classifier on train abalone

Elapsed time training: 0.03522491455078125

Evaluating Sci-kit GaussianNB classifier on test abalone

Elapsed Time predicting: 0.008008718490600586

Evaluation metrics for class: 0
Accuracy: 0.861244019138756
Precision: 0.4256410256410256
Recall: 0.9540229885057471
F1-score: 0.5886524822695035

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       83      |       4       | Positive
|---------------|---------------|              Actual
|       112     |       637     | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.5873205741626795
Precision: 0.8753117206982544
Recall: 0.5433436532507739
F1-score: 0.6704871060171919

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       351     |       295     | Positive
|---------------|---------------|              Actual
|       50      |       140     | Negative
|---------------|---------------|

Evaluation metrics for class: 2
Accuracy: 0.7188995215311005
Precision: 0.225
Recall: 0.5242718446601942
F1-score: 0.31486880466472306

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       54      |       49      | Positive
|---------------|---------------|              Actual
|       186     |       547     | Negative
|---------------|---------------|

mean accuracy: 0.583732057416268
```

Wine

```
Training NaifBayes classifier on train wine

Elapsed time: 0.03199172019958496

Evaluating BayesNaif classifier on test wine

Elapsed time prediction: 0.03400015830993652

Evaluation metrics for class: 0
Accuracy: 0.812962962962963
Precision: 0.889261744966443
Recall: 0.7957957957957958
F1-score: 0.8399366085578447

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       265     |       68      | Positive
|---------------|---------------|
|       33      |       174     | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.812962962962963
Precision: 0.71900826446281
Recall: 0.8405797101449275
F1-score: 0.7750556792873051

Confusion matrix
            Predicted
    Positive    |    Negative
|---------------|---------------|
|       174     |       33      | Positive
|---------------|---------------|
|       68      |       265     | Negative
|---------------|---------------|

mean accuracy: 0.812962962962963
```

```
Training scikit GaussianNB classifier on train wine

Elapsed time training: 0.02925562858581543

Evaluating Sci-kit GaussianNB classifier on test wine

Elapsed Time predicting: 0.007992744445800781

Evaluation metrics for class: 0
Accuracy: 0.8185185185185185
Precision: 0.8956228956228957
Recall: 0.7987987987987988
F1-score: 0.8444444444444443

Confusion matrix
    Predicted
Positive        |    Negative
|---------------|---------------|
|       266     |       67      | Positive
|---------------|---------------|
|       31      |       176     | Negative
|---------------|---------------|

Evaluation metrics for class: 1
Accuracy: 0.8185185185185185
Precision: 0.7242798353909465
Recall: 0.8502415458937198
F1-score: 0.7822222222222223

Confusion matrix
    Predicted
Positive        |    Negative
|---------------|---------------|
|       176     |       31      | Positive
|---------------|---------------|
|       67      |       266     | Negative
|---------------|---------------|

mean accuracy: 0.8185185185185185
```

Actual

Actual

Actual

Actual