UNIVERSITÉ DE SHERBROOKE

IFT 712 – Techniques d'apprentissage

Projet de fin de session

Professeur:

Martin Vallières

Réalisé par :

BOUFARIS Jinane 23 225 485
BOUSSAIRI Hamza 23 225 458
JAMALY Abir 23 225 494

Mail :

Jinane.Boufaris@USherbrooke.ca
Hamza.Boussairi@USherbrooke.ca
Abir.Jamaly@USerbrooke.ca

Année Universitaire 2023/2024

List of Figures

Contents

Introduction générale

Le but de ce projet est de comparer six classifieurs de la bibliothèque Scikit-Learn pour classifier des feuilles issues de différentes espèces de plantes. Dans ce rapport nous allons décrire en detail toutes les etapes du projet. Primo, nous allons établir la description de la base de données avec laquelle on travaille. Secundo, Nous alllons présenter le design de notre projet en expliquant les différentes classes crées. Par la suite, nous allons procéder à la description des divers classifieurs exploités en comparant les différents résultats obtenus après classification. Finalement, nous cloturerons notre rapport par les outils de gestion de projet que nous avons utilisé tout au long de notre travail.

Lien de notre travail: GitHub Repo

1 Description des données

Le jeu de données comprend environ 1584 images d'échantillons de feuilles, converties en feuilles binaires noires sur fond blanc. Pour chaque caractéristique, un vecteur de 64 attributs est donné par échantillon de feuilles.

Les attributs de notre jeu de données sont:

- id - Identifiant de l'image
- species - espèce de plantes à laquelle appartient l'image
- margin_1, margin_2, ..., margin_64 - chancun des 64 vecteurs d'attributs pour la caractéristique de marge
- shape_1, shape_2, ..., shape_64 - chacun des 64 vecteurs d'attributs pour la caractéristique de form
- texture_1, texture_2, ..., texture_2 - chacun des 64 vecteurs d'attributs pour la caractéristique de texture

Notre dataset contient 990 observations, 194 attributs, ainsi que 99 classes. Nous avons remarqué que les classes sont parfaitement balancés, contenant 10 observations chacune.

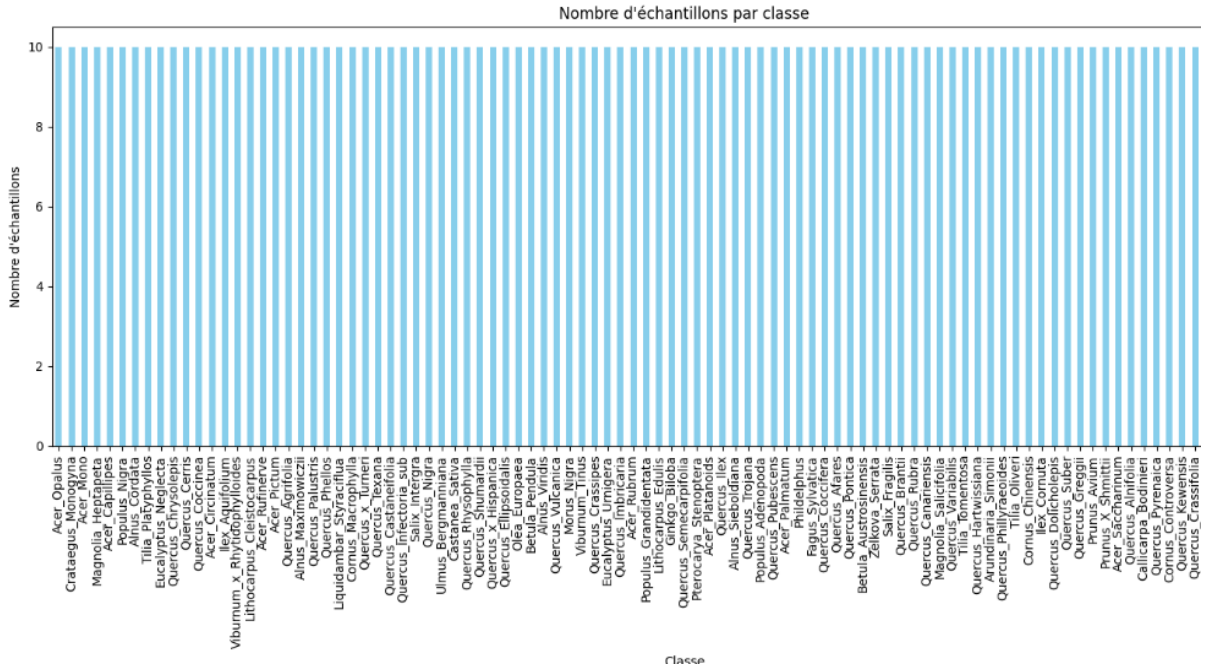

Figure 1: nombre d'echantillons par classe

2 Design du code

Nous avons choisi d'implementer trois différentes modules, comme le montre le diagramme suivant:

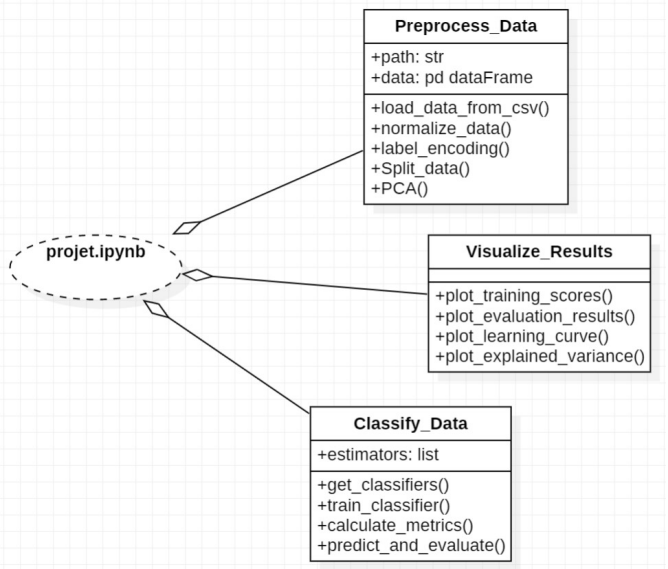

Figure 2: Diagrammes des classes

- Le module Preprocess_Data contient:
  - une fonction load_data_from_csv qui permet de charger les données en dataframe
  - une fonction normalize_data qui normalise les données en utilisant MinMaxScaler
  - une fonction label_encoding qui transforme les données catégoriques en données numériques
  - une fonction PCA qui applique le PCA sur notre dataframe afin de réduire ses dimensions
- Le module Classify_Data contient:
  - une fonction get_classifiers qui crée une lliste contenant les instances des classifieurs avec leurs hyperparametres
  - une fonction train_classifier qui entraine les classifiers et utilise le "grid search" pour trouver les meilleurs hyper-parametres de chaque classifieur
  - une fonction calculate_metrics qui calcule les metriques d'évaluation, notamment la justesse, le rappel, la precision et le f1 score.
  - une fonction predict_and_evaluate qui calcule les prédictions et retourne les résultats de l'evaluation en utilsant les métriques citées avant.

- Le module Visualize_Results contient:
  - une fonction plot_training_scores qui visualise les meilleurs scores de chaque classifieur après entrainement
  - une fonction plot_evaluation_results qui affiche les differents metrique d'evaluations pour chacun des classifieurs
  - une fonction plot_learning_curve qui afficher la courbe d'entrainement ainsi que la courbe de test de chaque classifieur
  - une fonction plot_explained_variance qui affiche la courbe de la variance expliquée

Dans ce projet nous avons utilisé principalement la bibliothèque Scikit-Learn. Aussi, nous avons adapté notre code d'une telle manière qu'il focntionne avec tous les algos quelque soit leurs hyper-paramètres.

3 Démarche scientifique

3.1 Pretraitement de donnees

Après avoir explorer notre jeu de données, nous avons constaté que nos données sont déjà pré-traitées. Ainsi, nous n'avons trouvé aucune donnée manquante. De plus, toutes les caractéristiques sont de type numérique. Aussi, nous avons remarqué que nos classes sont parfaitement balancées. De ce fait, notre prétraitement a consisté à:

- Standardiser nos données en utilisant le MinMaxScaler.
- Transformer les classes catégoriques en classes numériques en appliquant le "le label encoding"
- La séparation des données en deux parties; une pour l'entrainement des modèles et l'autre pour le tester
- Manipulation des caractéristiques en utilisant l'ACP pour réduire les dimensions de notre jeu de données.

3.2 Modèles utilisés

Nous avons utilisé exactement six modèles classifieurs, comme demandé:

- **Régression Logistique**: C'est un modèle linéaire qui se sert de la fonction logistique pour établir la relation entre les caractéristiques Xi et la cible Y. Autrement dit, il sert à prédire la probabilité qu'un événement arrive ou pas. Dans le cas multi-classes la régression logistique de la librairie scikit-learn utilise le "one-vs-rest" pour traiter plusieurs classes, en les considèrant individuellement par rapport aux autres.
- **SVC** : Support vector classifier est un modèle d'apprentissage machine supervisé utilisé pour la classification. Il consiste à trouver un hyperplan dans un espace à haute dimension qui sépare au mieux les points de données appartenant à différentes classes. Le SVC a pour objectif de maximiser la marge entre les classes, qui est la distance entre les vecteurs de support et la surface de séparation, tout en minimisant les erreurs de classification. Ce modèle est performant dans le traitement des données linéaires et non linéaires à l'aide des fonctions à noyau.
- **SGD** : C'est un modèle qui utilise la descente de gradient stochastique pour l'apprentissage. il se met à jour de manière itérative, ce qui permet un apprentissage par mini-batch et fonctionne avec des données denses ou dispersées. Il permet d'utiliser diverses fonctions de perte et méthodes de régularisation (L1, L2) pour éviter l'overfitting.

- **Random Forest** : Ce modèle combine un ensemble d'arbres de décision et réduit leur variance pour obtenir des performances de généralisation plus intéressantes. Chaque arbre de décision est établi en utilisant un sous-ensemble aléatoire des données d'apprentissage et un sous-ensemble aléatoire des paramètres, ce qui permet de faire varier les arbres. Cette approche permet de réduire le sur-apprentissage et d'améliorer les performances de généralisation.
- **Adaboost** : Adaptive Boosting est un modèle qui combine une série de modèles avec une faible capacité, géneralement des arbres de décision simple, sur des sous-ensembles de données d'apprentissage. À chaque itération, l'algorithme attribue des poids plus élevés aux points de données mal classés, se concentrant ainsi sur ces points difficiles à classer correctement. La prédiction finale est obtenue par une combinaison pondérée de ces modèles.
- **Bagging** : Bootstrap Aggregating consiste à combiner plusieurs instances du même modèle d'apprentissage de forte capacité, sur différents sous-ensemble de données d'apprentissage qui sont génerés par le biais du bootstrapping. Chaque modèle est entraîné de manière indépendante et effectue ses prédictions. La prédiction finale est ensuite déterminée par un vote, pour la classification. Le bagging permet de réduire le sur-apprentissage et la variance en exploitant la diversité des modèles entrainés indépendamment les uns des autres.

3.3 Définition des classifieurs

Nous avons défini les modèles classifieurs choisies dans une liste de dictionnaires comme suit:

```
six_classifiers = [
    {
        'model':'RandomForestClassifier',
        'params': {
            'n_estimators' : [50, 75, 100],
            'min_samples_split': [2, 3, 4, 5],
            'criterion': ["gini", "entropy"],
        }
    },

    ...

    {
        'model': 'LogisticRegression',
        'params':{
            'C': [0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 1.5, 2.0],
            'solver': ['liblinear', 'newton-cg', 'lbfgs']
        }
    }
]
```

Figure 3: structure de la liste des classifieurs

La figure précédente représente une partie de la liste des six classifieurs avec lesquelles nous avons travaillé tout au long de notre projet. La liste contient un ensemble de dictionnaires, chacun définit un des six classifieurs utilisés, la valeur de la clé 'model' n'est rien d'autre que le nom du modèle. La clé 'params' a comme valeur un dictionnaire qui définit les hyper-paramétres, parmi lesquels nous choisisssons les meilleurs en utilisant la méthode de GridSearch.

Pour pouvoir exploiter ces différents dictionnaires, dans la GridSearch nous aurons besoin des instances des classifieurs (en effet, les dictionnaires ne contiennent que les noms des modèles et leurs hyper-parametres). De ce fait, nous utilisons la fonction get_classifiers qui parcoure la liste des classifieurs et ajoute, à chaque dictionnaire, une clé 'classifieur' qui prend comme valeur l'instance associée au modèle.

Et c'est cette nouvelle liste que reçoit la fonction train_classifier en entrée, pour effectuer la GridSearch.

3.4 GridSearch et Validation Croisée

Le GridSearchCV est un outil puissant utilisé pour optimiser les hyperparamètres d'un modèle. Le GridSearchCV, qui provient de la bibliothèque scikit-learn, ef-

fectue une recherche exhaustive dans une grille d'hyperparamètres pré-définie par l'utilisateur.

Il évalue les performances du modèle pour chaque combinaison d'hyperparamètres en utilisant la validation croisée; et ceci en divisant l'ensemble de données en plusieurs plis et évaluer le modèle sur chacun d'eux. Cela garrantit que le modèle est évalué sur les différents portions des données, ce qui lui permet de mieux généraliser.

Dans notre cas, nous avons définit une grille d'hyperparamètres pour chaque modèles et, à l'aide du GridSearchCV, nous avons recherché la meilleur combinaison des valeurs qui maximise la performance des modèles.

4 Résultats et analyse

4.1 Métriques utilisées

Pour évaluer nos classifieur et pouvoir les comparer, nous avons utilisé les métriques suivantes :

- **Accuracy** : C'est la proportion de la prédictions correctes dans un ensemble de données. Elle présente le rapport entre le nombre de prédictions correctes et le nombre total des données.

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{\text{True positives} + \text{True negatives} + \text{False positives} + \text{False negatives}}$$

- **Recall** : Mesure la capacité du modèle à prédire les classes positives réelles. Il s'agit du rapport entre les vrais poisitifs prédits et ce qui est réellement positifs.

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

- **Precision** : Mesure l'exactitude du modèle en évaluant le rapport entre le nombre de vrais positifs et le nombre total des positifs prédits.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

- **f1_score** : S'agit de la moyenne harmonique de la précision et du rappel. Il est utile lorsqu'on recherche un compromis entre la precison et le Recall

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

On a choisi cette combinaison de métriques pour fournir un évaluation complète de nos modèles. L'Accuracy donne une vue d'ensemble, le Recall mesure la capacité à détecter les instances positives, la Precision est cruciale lorque les faux positifs ont un impact significatif et le f1-score combine precision et recall pour une évaluation globale équilibrée. Ces mesures nous ont permis de comprendre en profondeur les performances et la qualité des prédictions de nos modèles.

4.2 Visualisations et interpretation

Après avoir calculé les différentes métriques d'évaluation pour chaque modèle, nous avons décidé d'afficher nos résultats sous forme de diagrammes à bandes pour les rendre plus lisibles et comprhénsibles.

4.2.1 Sans modification des dimensions

La figure ci-dessus représente la métrique de la justesse. Les autres visualisations faites seront mises en annexe.

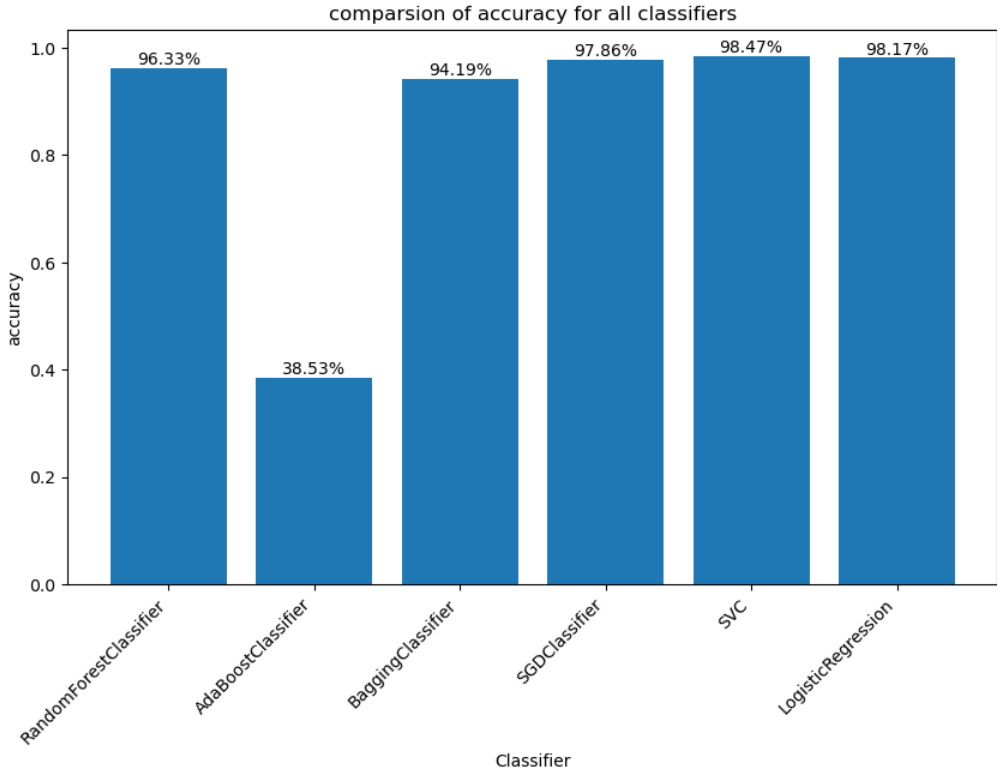

Figure 4: Comparaison de la justesse des classifieurs

Nous remarquons que cinq parmi six classifieurs ont été performants en terme de toutes les métriques. Ils possèdent tous des scores dépassants 90% , sauf pour le AdaBoostClassifier qui a eu de faible performances pour toutes les métriques.

D'après la visualisation de toutes les métriques nous pouvons constaté que la régression logistique est la plus perfomante dans notre cas, suivi du SVC et SDG-Classifier. En fait, les résultats de tous les classifieurs sont proches et peu similaires.

Nous avons aussi pensé à évaluer le sur-apprentissage et le sous-apprentissage en affichant les courbes d'apprentissage de nos classifieurs. (Voir l'Annexe)

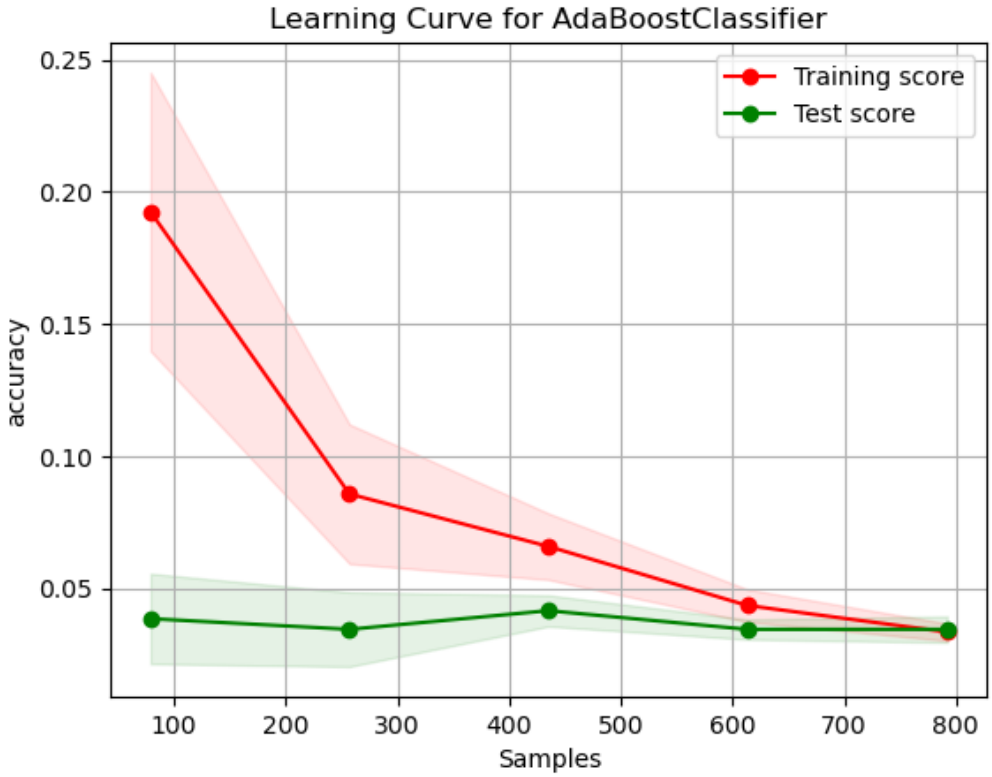

Figure 5: Courbe d'apprentissage de Adaboost

Nous concluons, tout de suite, que Adaboost est en sous-apprentissage, en regardant sa courbe d'entrainement qui indique que le modèle apprend mal l'ensemble des données d'apprentissage.

Pour les autres modèles, nous remarquons qu'ils performent bien sur les deux ensembles d'entrainement et de test.

Pour voir si nos résultats sont pertinents ou pas, nous nous sommes référés à des résultats de test d'un travail déjà présent en ligne sur le site du challenge. C'est un travail qui vise la comparaison de 10 classifieurs, dont trois sont identiques à ceux que nous avions utilisés (SVC, AdaBoost, RandomForest). Ainsi, nous pouvons remarquer que nos résultats sont plus ou moins proche de ceux du travail sur Kaggle. Nous pourrons meme dire que nos modèles sont plus performants avec RandomForest à 96.8% contre 88.8%, SVC à 98.1% contre 81.8% et Adaboost 34.9% contre 4.54%.

Pour résumer, nous avons obtenu des résultats plus favorables que ceux sur kaggle, peut-etre en raison de l'utilisation de GridSearchCv pour determiner les hyperparamètres optimaux, ce qui n'a pas été le cas dans leur approche.

4.2.2 En utilisant ACP

L'Analyse en Composantes Principales (ACP) a été utilisée pour réduire la dimensionnalité des données d'entraînement, permettant ainsi de résumer l'information

contenue dans un grand nombre de colonnes tout en éliminant celles ayant moins d'importance. L'application de la méthode du coude (elbow method) a indiqué que le choix de 40 (au lieu de 990) capture 99% de l'information initiale, facilitant ainsi un entraînement plus rapide tout en conservant une représentation significative des données.

Cette approche a permis d'améliorer la performance de modèles tels que le Random Forest et le Bagging Classifier dans toutes les métriques utilisées, y compris le F1-score, la précision, le rappel et la justesse. En revanche, le SGD Classifier a montré une diminution de performance après l'ACP, soulignant l'importance de considérer la nature spécifique des algorithmes lors de l'utilisation de techniques de réduction de dimensionnalité.

5 Gestion de Projet

5.1 Git et Github

Travailler en équipe nous impose de collaborer tous pour établir un meme code qui fonctionne. Ainsi, nous étions obligés d'utiliser un outil de controle de version, pour pouvoir modifier notre code commun sans aucun conflits. Nous avons éviter les mauvais pratiques, comme demandé dans l'énoncé du projet. Ainsi, chacun, de nous, faisait des modifications sur le code dans une branche différente de la branche master. Ensuite, il effectue un "pull-request", nous vérifions tous le code proposée pour enfin décider de merger la branche concernée au master.

5.2 Trello

Pour mener un travail bien organisé et structuré, nous distribuions les taches équitablement sur tout les membres de l'équipe. Ceci ne peut se faire correctement qu'à l'aide d'un outil de gestion de projet approprié, notamment Trello. Nous avons procédé par créer des tickets qui correspondent chacune à une tache particulière. Les taches passent de 3 états: "To Do" , "In progress" et "Done". Ainsi, tous les membres de l'équipe peuvent avoir une idée sur l'avancement des taches des autres.

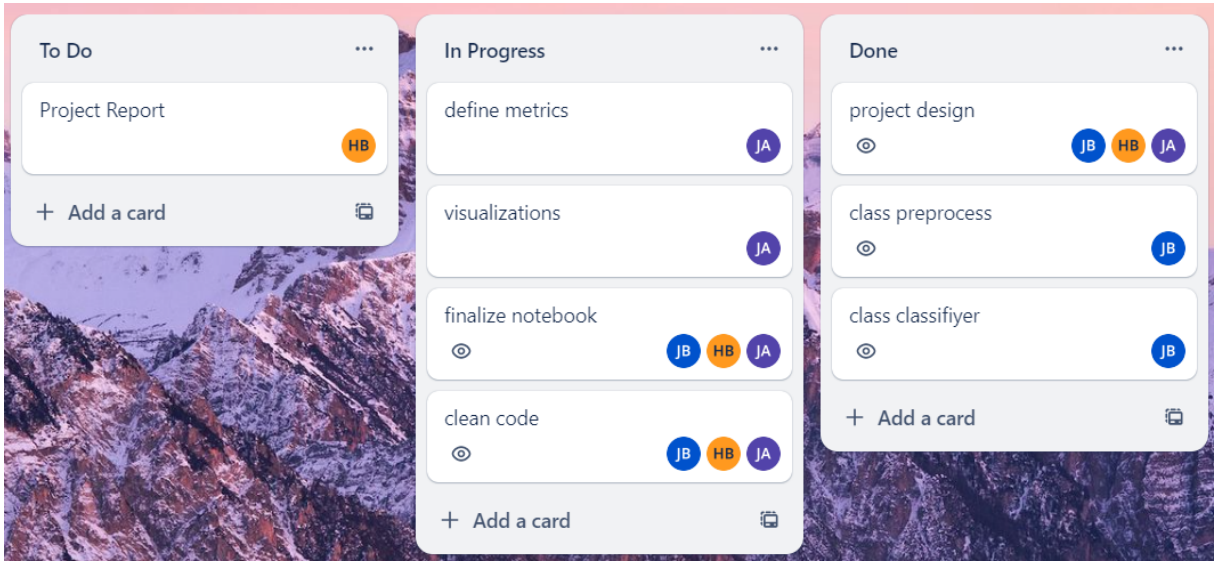

Figure 6: Utilisation de Trello pour distribution des taches

6 Annexe

6.1 Scores d'entrainement

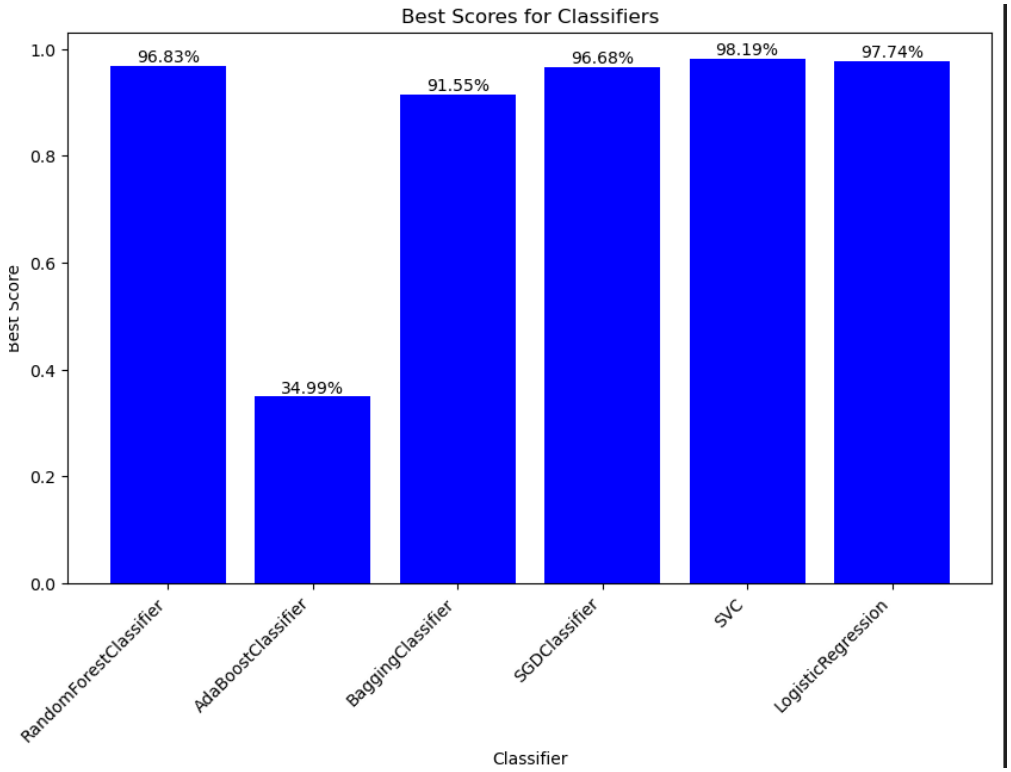

Figure 7: Scores d'entrainement sans reduction de dimensions

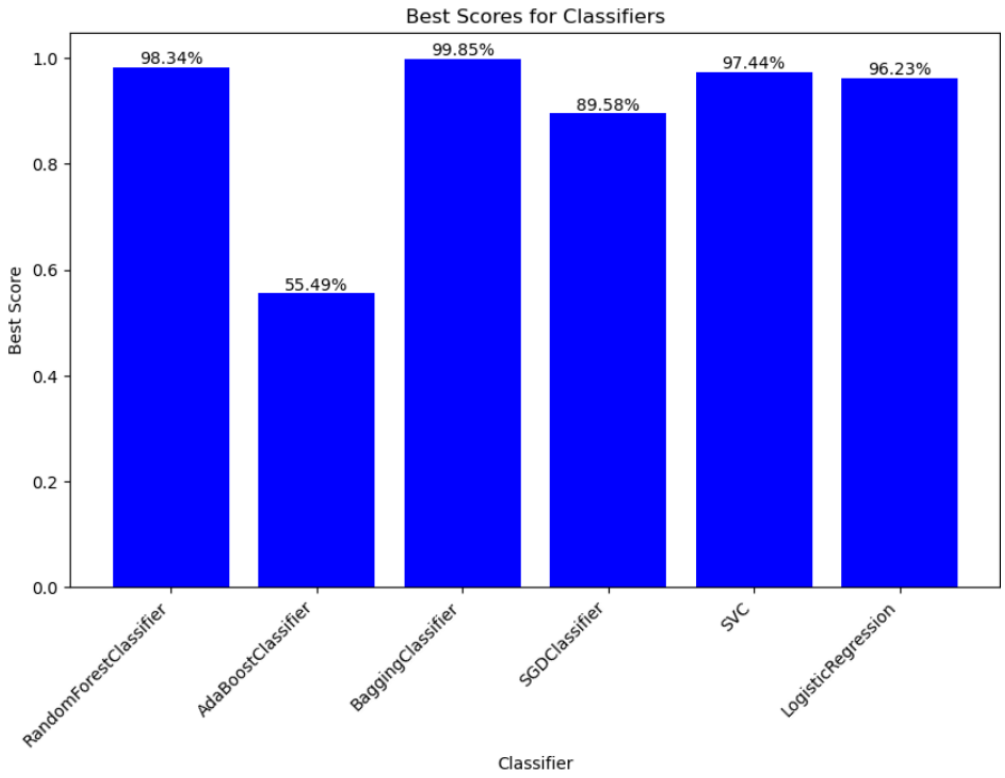

Figure 8: Scores d'entrainement en utilisant ACP

6.2 Métriques d'évaluation pour l'ensemble de Test

6.2.1 Avant reduction de dimensions

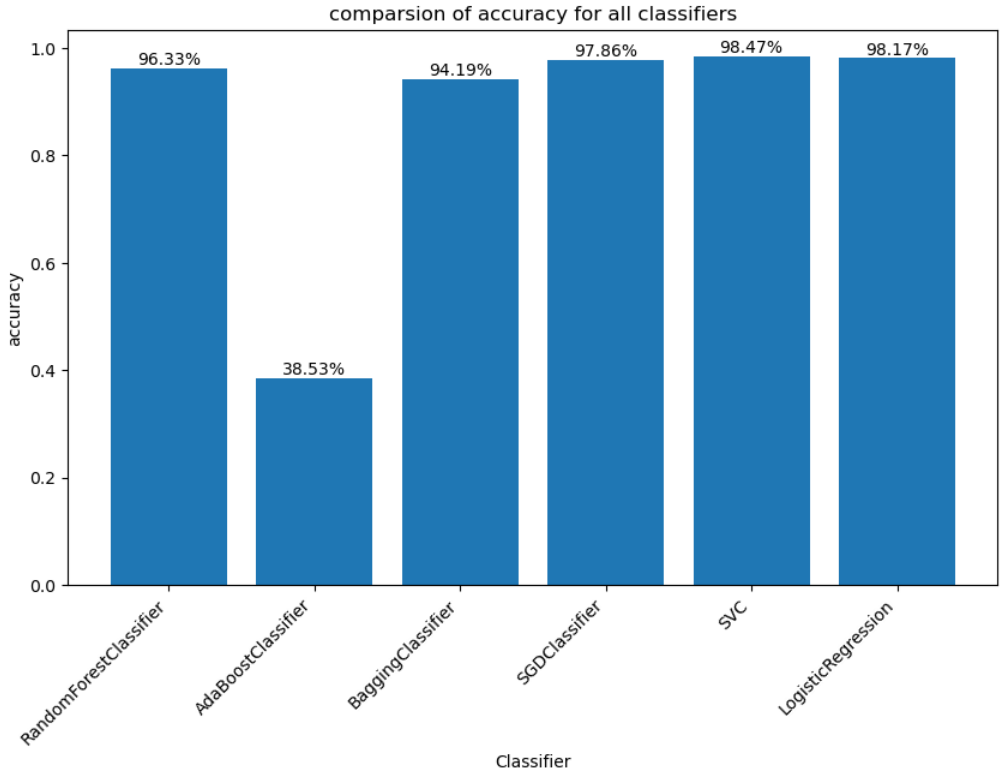

Figure 9: Justesse

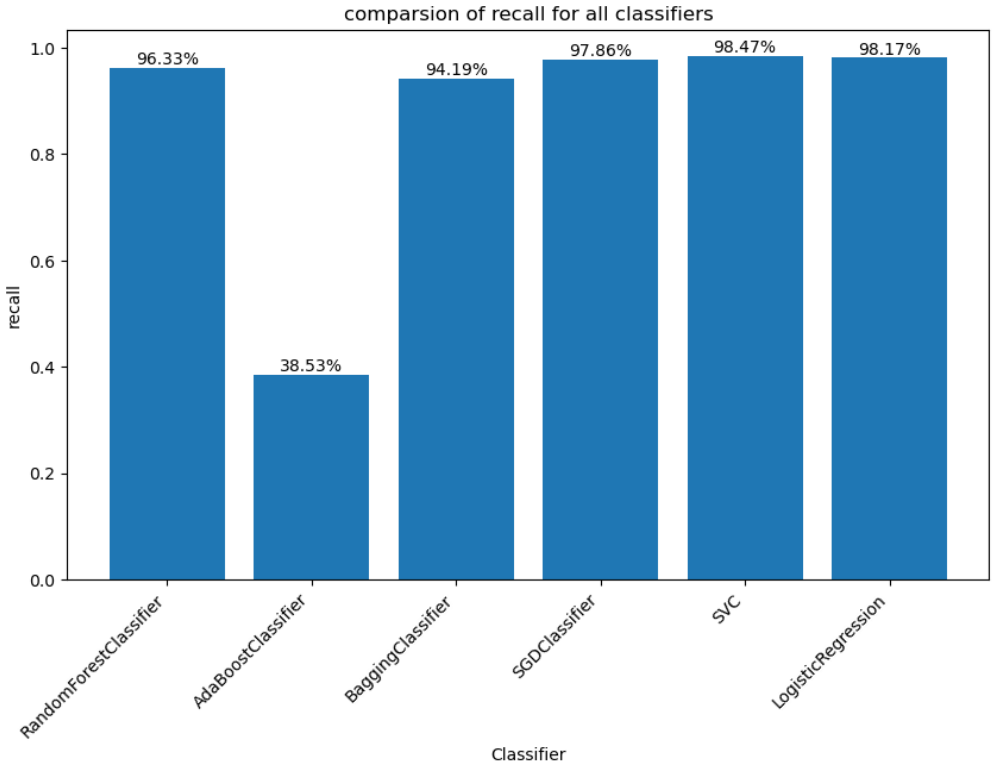

Figure 10: Rappel

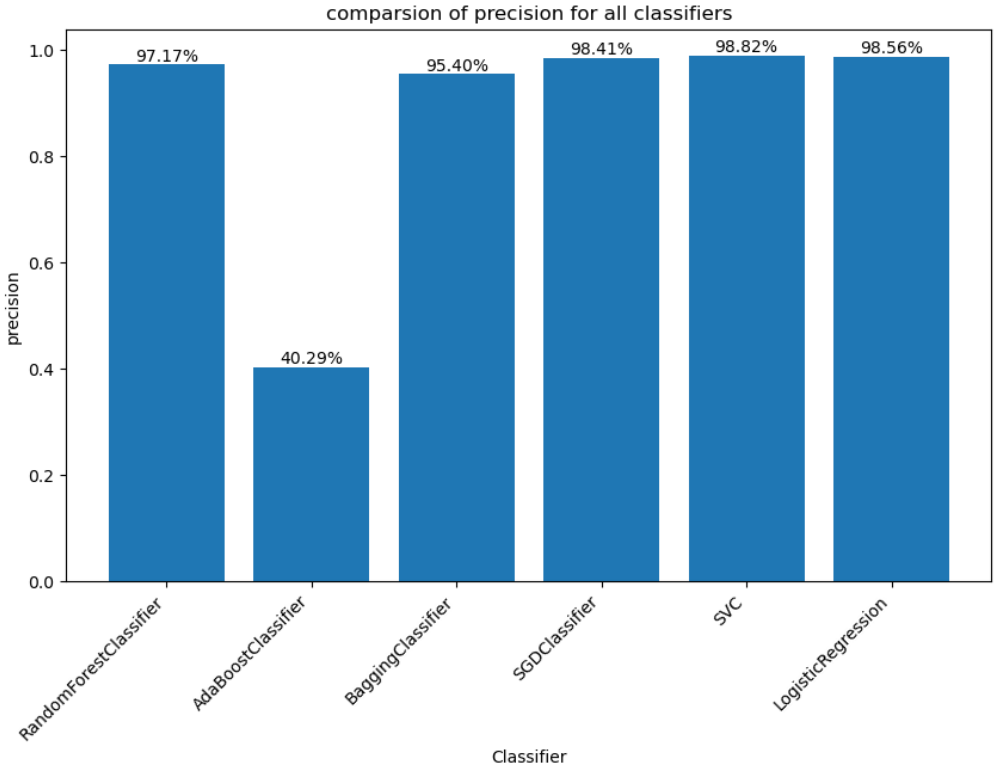

Figure 11: Precision

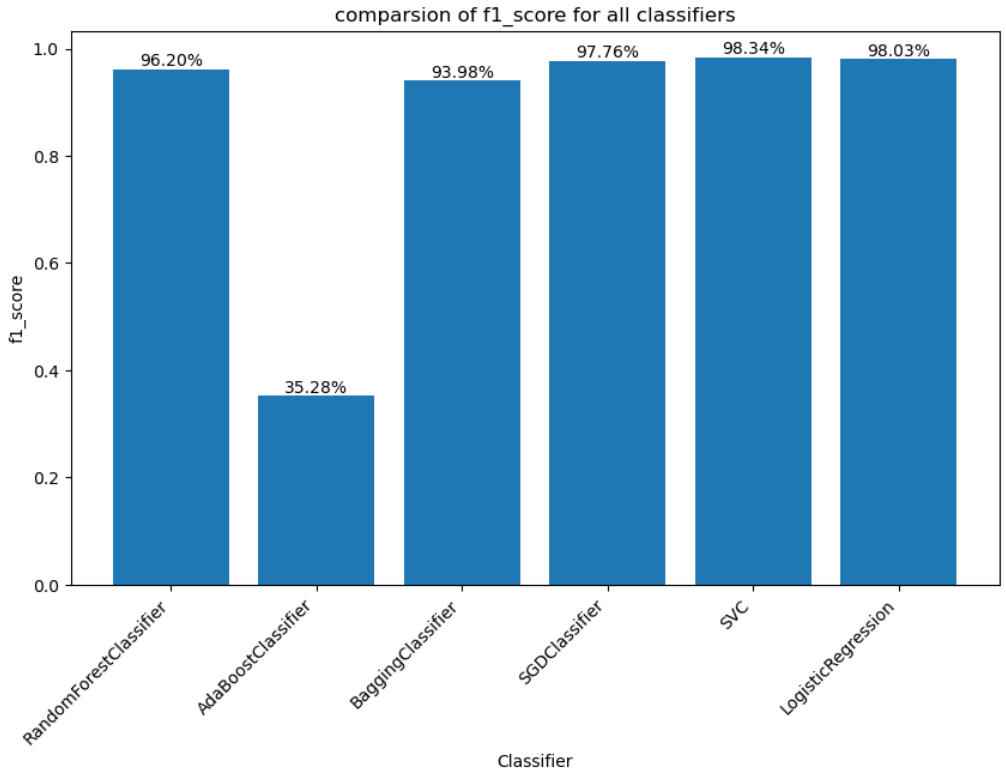

Figure 12: F1 score

6.2.2 En utilisant ACP

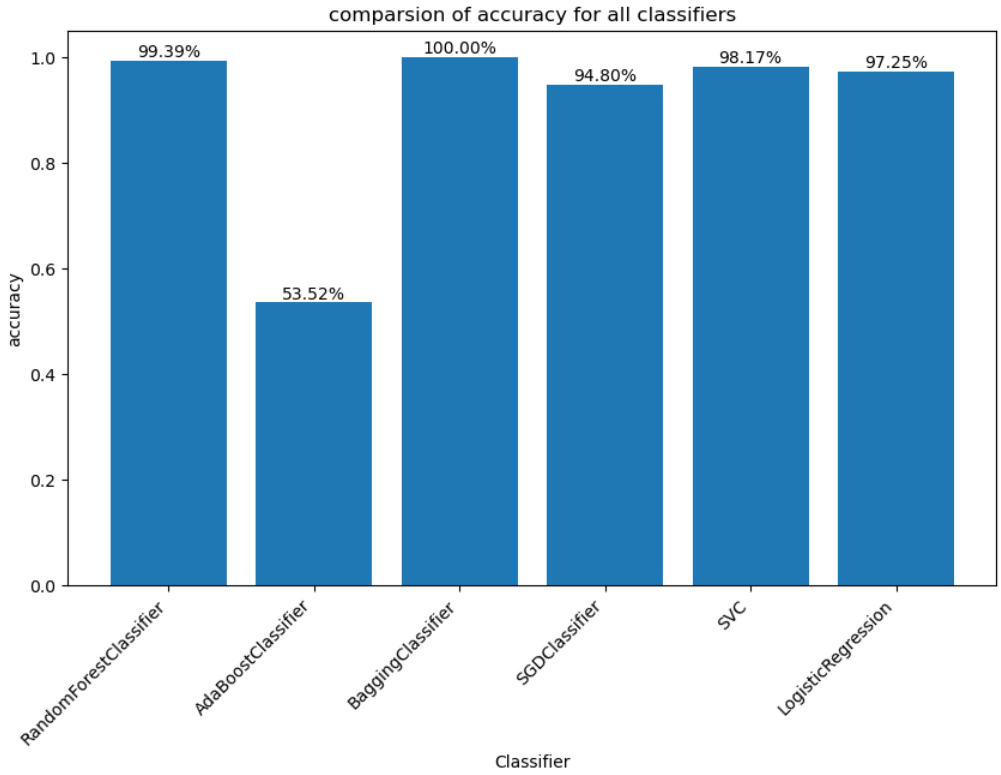

Figure 13: Justesse

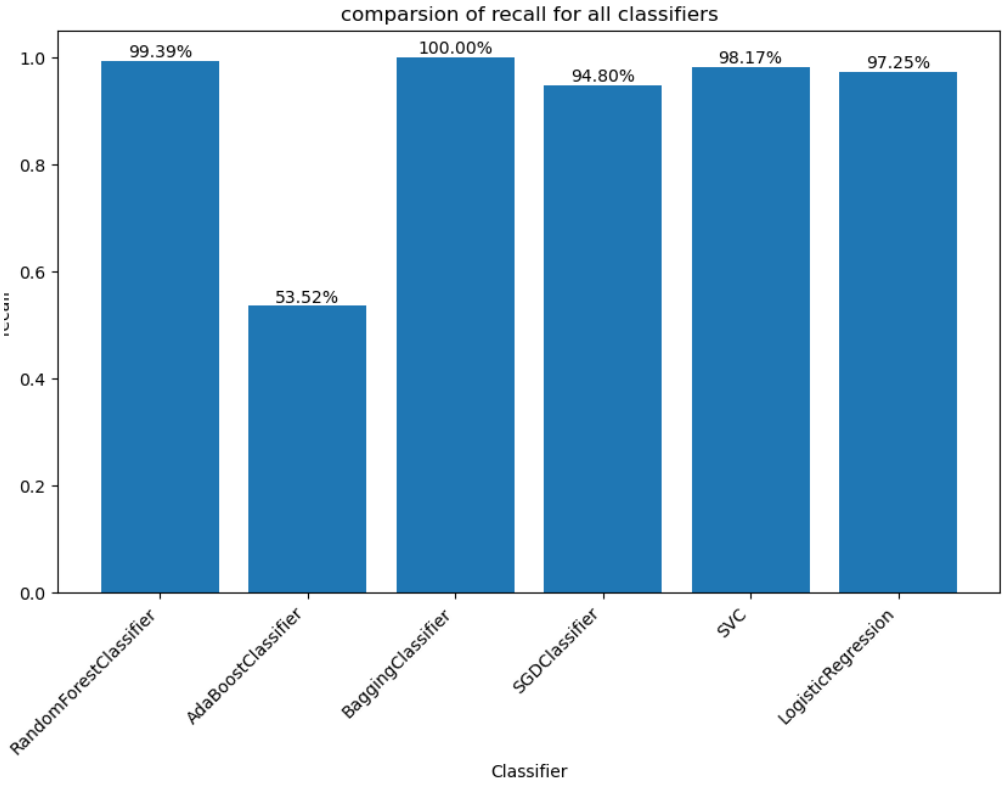

Figure 14: Rappel

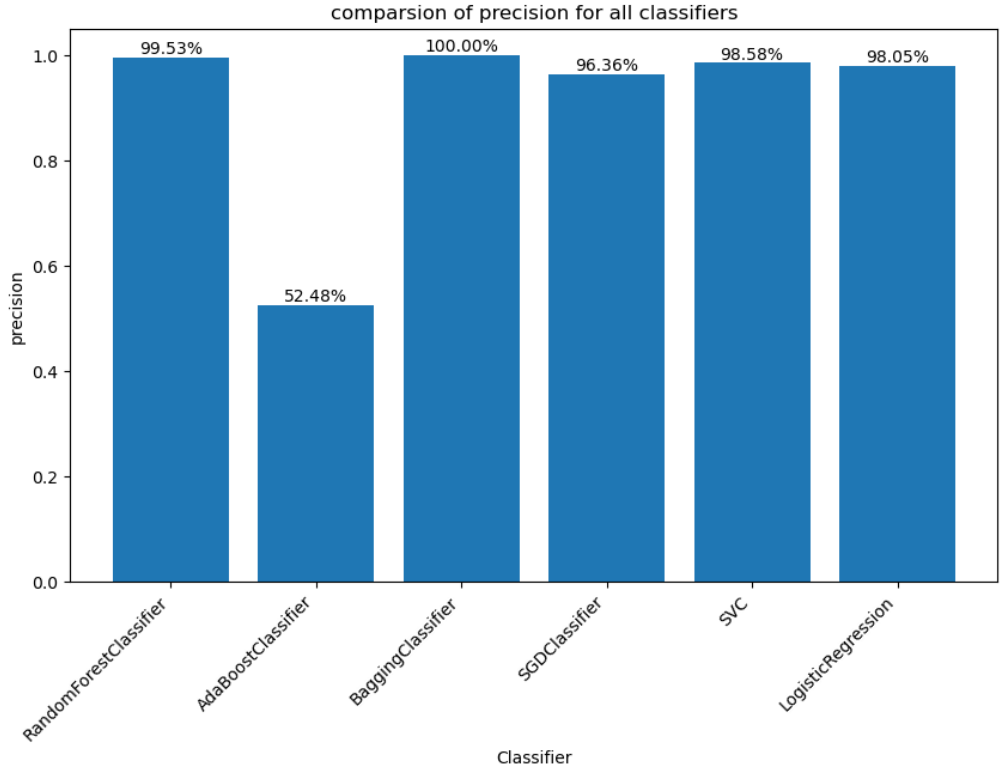

Figure 15: precision

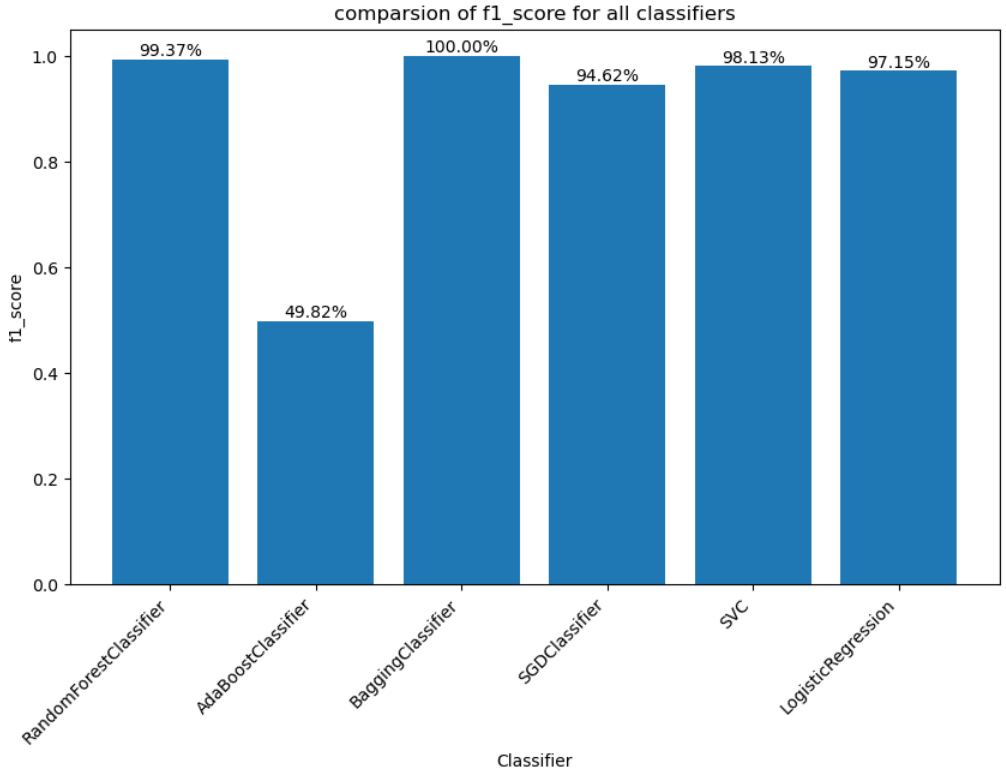

Figure 16: f1_score

6.3 Courbes d'apprentissage

6.3.1 Avant reduction de dimensions

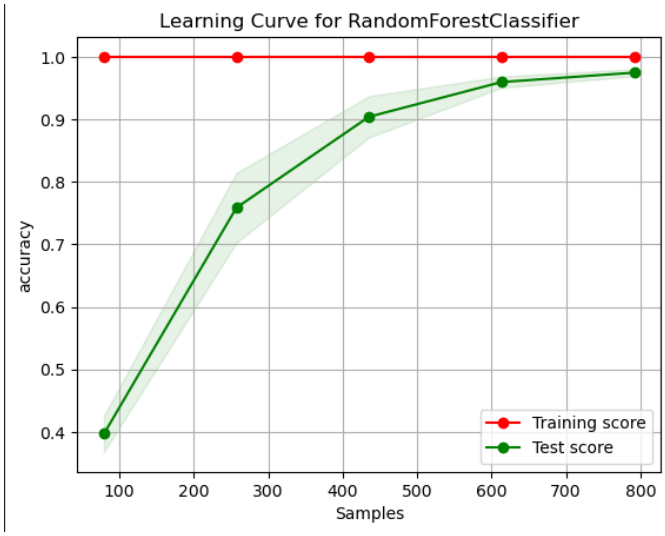

Figure 17: Courbe d'apprentissage pour Randomforest

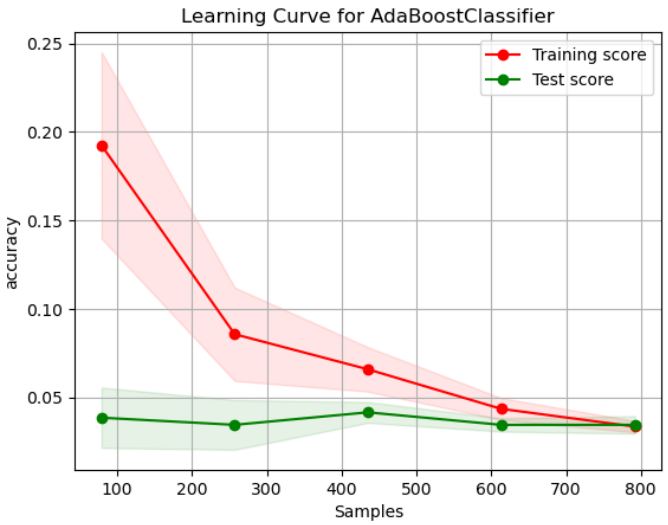

Figure 18: Courbe d'apprentissage pour Adaboost

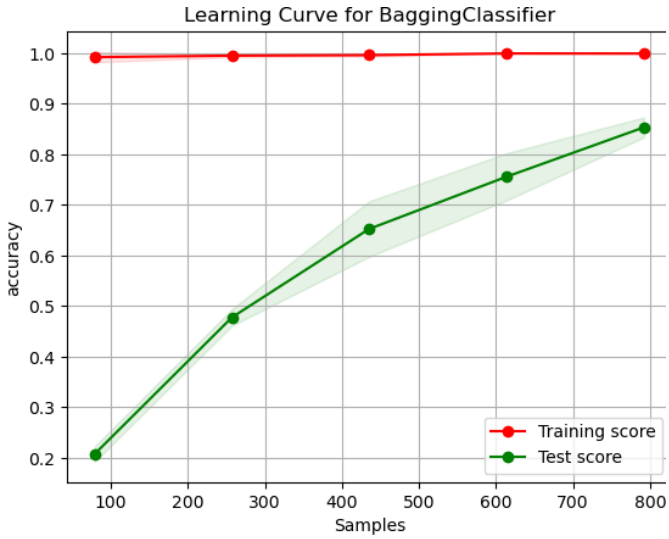

Figure 19: Courbe d'apprentissage pour Bagging

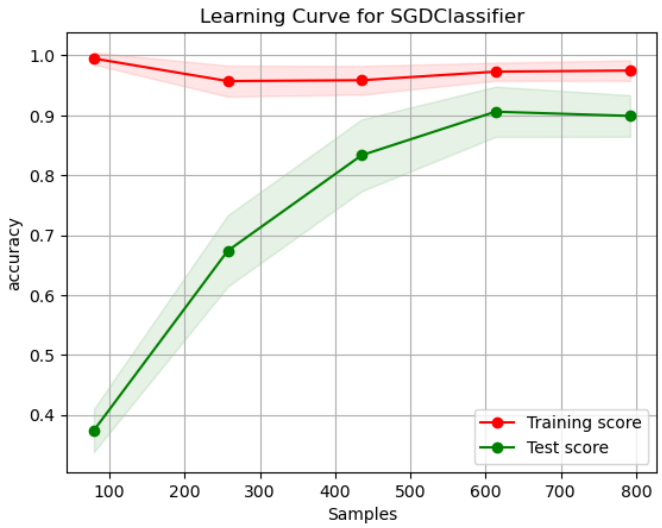

Figure 20: Courbe d'apprentissage pour SGD

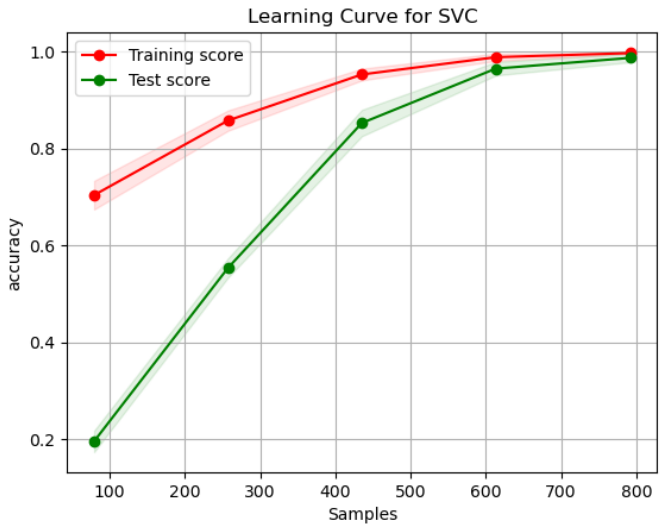

Figure 21: Courbe d'apprentissage pour SVC

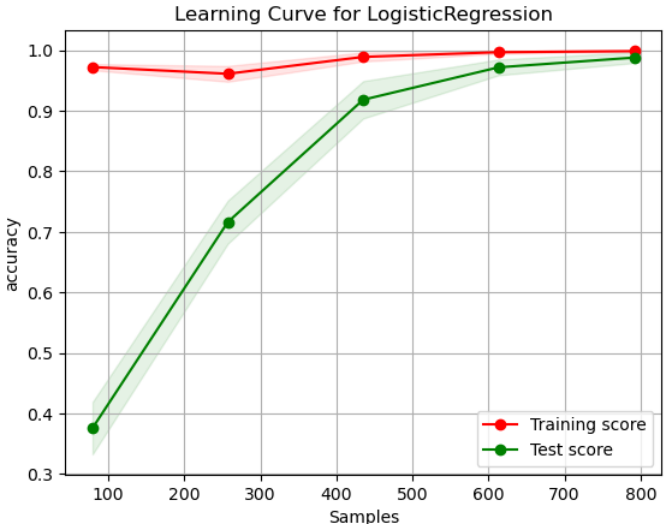

Figure 22: Courbe d'apprentissage pour logistiRegression

6.3.2 Apres ACP

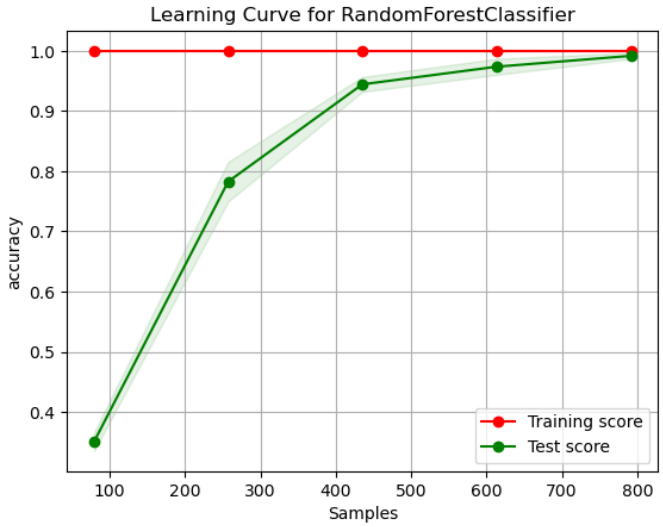

Figure 23: Random forest

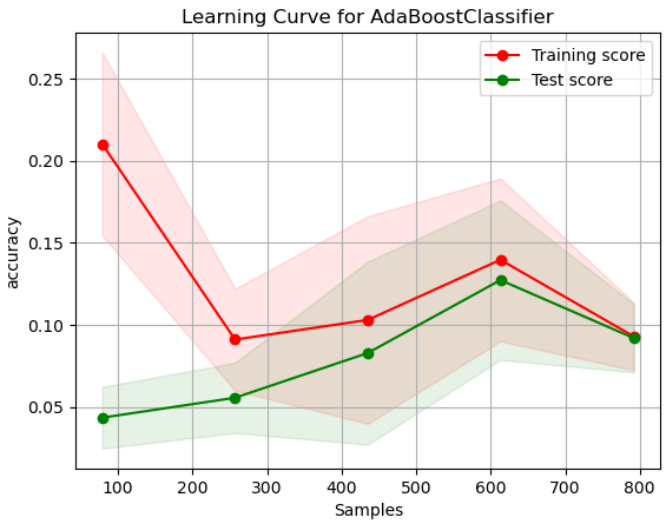

Figure 24: Adaboost

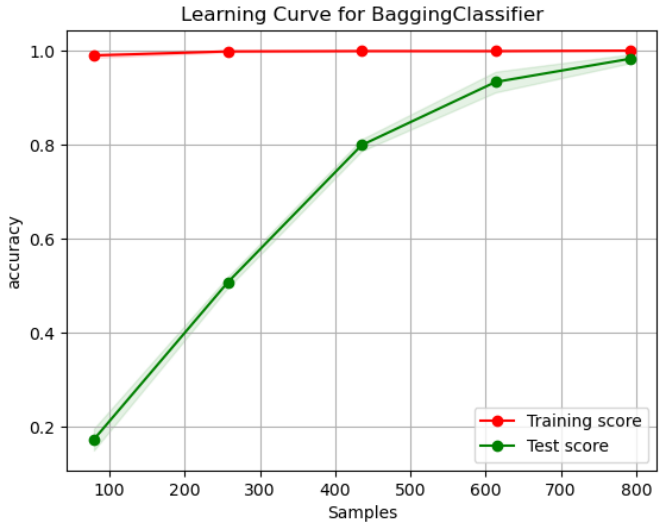

Figure 25: Bagging

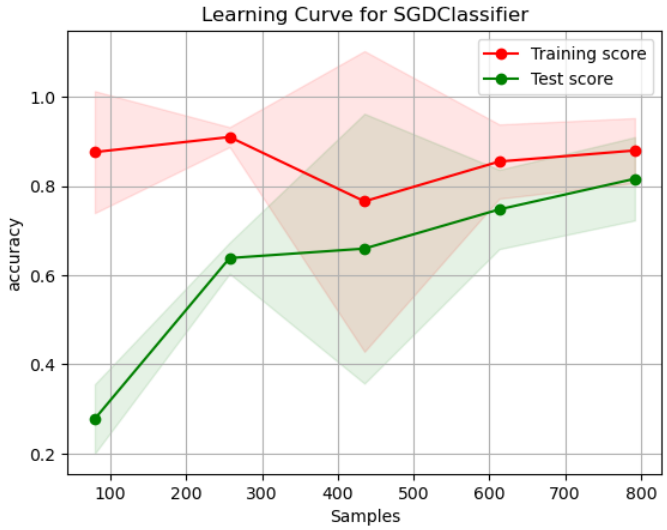

Figure 26: SGD

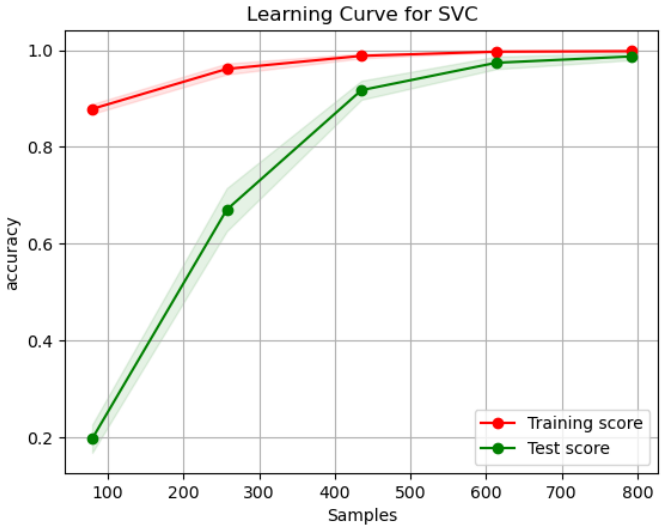

Figure 27: SVC

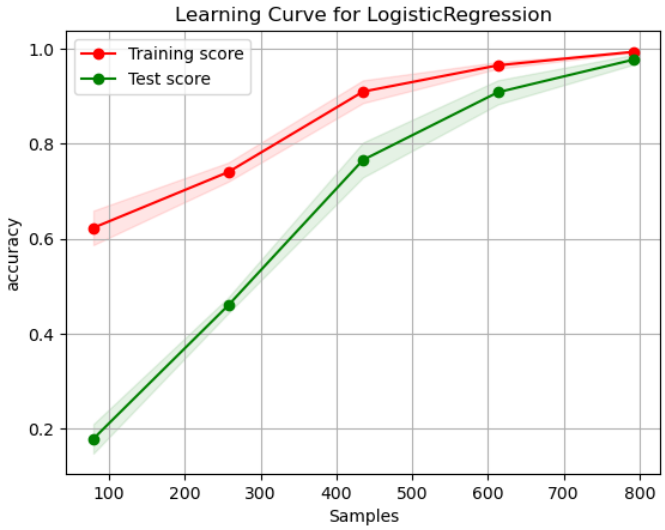

Figure 28: Logistic regression

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html.

[8] https://learn.microsoft.com/fr-fr/azure/ai-services/language-service/custom-text-classification/concepts/evaluation-metrics.