|  |  |  |  |
| --- | --- | --- | --- |
| **Machine Learning (ML)**  **Examen Final**  **1h30** |  | * Documents autorisés * Il faut rendre :   + page 1 de l’énoncé (cette feuille) ;   + une réponse sur papier (TP) ;   + un fichier Jupyter notebook avec votre nom. | |
| **Solution et barème** | | | Equipe ML |

**Questions (6pts)**

Q1) Il faut transformer les caractéristiques nominales en numériques avant d’appliquer **(1pt)**

Régression logistique  Naïve Bayes  Arbre de décision  SVM

Q2) Quels sont les algorithmes d'apprentissage qui peuvent classer le dataset suivant parfaitement **(1pt)**

|  |  |
| --- | --- |
| y  x | Régression linéaire  Régression logistique  Arbre de décision  SVM |

Q3) L’erreur de classification tend vers 0 lorsque le nombre des échantillons d’entrainement tend vers l’infini. Quand est-ce que cette déclaration est fausse ? **(1pt)**

|  |
| --- |
| Une des deux réponses :   * Lorsqu’il n’y a pas de diversité dans les données (répétition) * Lorsque l’algorithme n’est pas adéquat pour le problème (comme la question Q2) |

Q4) Dessiner les graphes de performance d’un algorithme de classement non complexe d’entrainement et de test selon le nombre des itérations dans les deux cas : normal et sur-apprentissage **(1pt)**

|  |  |
| --- | --- |
| **performance**  **# itérations itérations**  **entraînement**  **test** | **performance**  **# itérations itérations** |
| Apprentissage normal | Apprentissage avec sur-apprentissage |

***La courbe descendante c’est l’erreur de classification, pour les gens qui ont fait ça (0.75pt)***

Q5) Dans les SVM, les classes sont +1 et -1. Que signifie 0 ? **(0.5pt)** Quel est l’intérêt ? **(0.5pt)**

|  |
| --- |
| * 0 signifie que l’échantillon n’appartient à aucune classe * L’intérêt est de séparer les classes avec une marge en maximisant cette dernière |

Q6) Vrai ou Faux (1pt)

a) Un classifieur entrainé sur moins de données est moins probable de sur-apprendre  Vrai  Faux

b) La régulation L1 peut être utilisée pour la sélection d’attributs  Vrai  Faux

c) Les arbres de décision n’ont pas besoin de sélection d’attribut  Vrai  Faux

d) On peut utiliser la régression logistique simple pour le classement multinomial  Vrai  Faux

**TP (14pts)**

* ***Normalement, il faut rendre Jupyter notebook. Pas de sanction.***
* ***Si une instruction contient une variable non définie, la note est 0. Dans cet examen le but est de juger la capacité de réutiliser des instructions déjà existantes dans un contexte défini et pas savoir quelle instruction à utiliser (puisque documents autorisés)***
* ***Si un traitement n’est pas utilisé par la suite, la note est 0. On juge le Pipeline et pas des choix multiples par l’étudiant (copier-coller le maximum et stocker les résultats dans des variables sans décider quelle variable à utiliser)***

**1- Préparation des données**

1.a- Lire le fichier, afficher son contenu et prétraiter les données sur le notebook **(2pts)**

|  |
| --- |
| ***animaux = pandas.read\_csv("animaux.csv", skipinitialspace=True) # ===> (0.25)***  ***animaux # ===> (0.25)***  ***r = animaux[animaux.duplicated("nom", keep=False)] # vérification des redoublants***  ***animaux.drop\_duplicates("name", keep="last", inplace=True) # ===> (0.5)***  ***u = animaux.backbone.unique() # vérification des valeurs de l'attribut "epine"***  ***animaux.drop(["backbone"], axis=1, inplace=True) # ===> (0.5)***  ***animaux.drop(["name"], axis=1, inplace=True) # ===> (0.5)*** |

***P.S. Normalement la vérification doit être présente. Mais, puisque nous avons oublié de le mentionner dans la question, sa note est attribuée au traitement***

1.b- Quels traitements avez-vous utilisé et pourquoi ? **(1pt)**

* Suppression des redoublants **(0.25)** puisque un échantillon « éléphant » est redoublé **(0.25)**
* Suppression des colonnes inutiles **(0.25)** puisque les colonnes « backbone » et « name » ne sont pas utiles pour le classement **(0.25)**

**2- Encodage des attributs**

2.a- Encoder les attributs afin d’utiliser les données par naïve Bayes multinomial, arbre de décision et la régression logistique **(1pt)**

|  |
| --- |
| ***# 0.5 (On peut utiliser OneHotEncoder aussi)***  ***animaux = pandas.get\_dummies(animaux, columns=["covering"]***  ***# ===> 0.5 pour positionner la classe en dernier ou pour ajouter l'encodage***  ***animaux["classe2"] = animaux["class"] animaux.drop(["class"], axis=1, inplace=True)*** |

2.b- Justifier le choix d’encodage **(1pt)**

* scikit-learn utilise des attributs numériques **(0.25)**
* Dans les arbres de décision (CART, utilisé par scikit-learn) et régression logistique on ne peut pas encoder les caractéristiques en entiers ; ceci peut favoriser unes des autres **(0.5)**
* Naïve Bayes nominal dans scikit-learn pour chaque caractéristique, on fournit le nombre de son occurrence. Dans ce cas, chaque valeur nominale d’une caractéristique peut être vue comme caractéristique avec une occurrence 0 ou 1 d’où l’encodage OneHot **(0.25)**

**3- Échantillonnage et division des données**

3.a- Diviser le dataset en données d’entraînement et de test en gardant les 4 derniers échantillons pour le test. **(1.5pt)**

|  |
| --- |
| ***X = animaux.iloc[:, :-1].values # Premières colonnes ===> 0.25***  ***Y = animaux.iloc[:,-1].values # Dernière colonne ===> 0.25***  ***# diviser X et Y sur test et entraînement ===> 0.5***  ***# prendre 4 échantillons pour test ===> 0.25***  ***# Prendre les derniers échantillons pour test ===> 0.25***  ***X\_train = X[:-4, :]***  ***X\_test = X[-4:, :]***  ***Y\_train = Y[:-4]***  ***Y\_test = Y[-4:]*** |

3.b- En se basant sur cette division :

* Est-ce que le dataset d’entraînement est déséquilibré ? Justifier la réponse en indiquant la classe majoritaire en cas de déséquilibre **(0.5pt) :** Oui, il est déséquilibré puisque le nombre des classes se défère (il y a des classes avec plusieurs échantillons par rapport aux autres)
* Mentionner une méthode pour régler le problème de déséquilibre **(0.5pt) :** Sur-échantillonnage, sous-échantillonnage, …
* Est-ce que cette division est raisonnable et pourquoi ? **(0.5pt).** Oui (il y a des cas : non). Concernant le pourcentage : 4/11 = 36% de test et 64% d’entrainement. En plus, le test contient toutes les classes (ceci est juste pour les gens qui n’ont pas randomisé ou trié par « name »). Dans les deux cas, il faut juger le nombre et la distribution des classes dans le test.

**4- Entraînement**

4.a- Entraîner trois modèles : naïve Bayes multinomial (NB), arbre de décision (CART) et la régression logistique (REG) en utilisant leurs implémentations dans scikit-learn **(0.75pt)**

|  |
| --- |
| ***naive\_bayes = MultinomialNB()***  ***naive\_bayes.fit(X\_train, Y\_train) # ===> (0.25)***  ***cart = DecisionTreeClassifier()***  ***cart.fit(X\_train, Y\_train) # ===> (0.25)***  ***reg = LogisticRegression(multi\_class="multinomial", solver="lbfgs")***  ***reg.fit(X\_train, Y\_train) # ===> (0.25)*** |

4.b- Après l’entraînement :

* Le F1 score sur l’entraînement *F1(NB, CART, REG) = (0.714, 1.0, 0.857)*. Qu’est-ce que chaque algorithme cherche à faire ? C.-à-d., comment une classe est vue par rapport aux caractéristiques ? **(1.5pt)** Pourquoi CART a pu atteindre 1.0 ? **(0.25pt)**
  + Naïve Bayes essaye de trouver les valeurs des caractéristiques dominantes dans des classes pour faire la séparation **(0.5)**
  + CART essaye de représenter la procédure de classement sous forme des règles (SI-SINON) sur les caractéristiques où le résultat est la classe **(0.5)**
  + La régression logistique essaye de séparer entre les classes linéairement **(0.5)**
  + Lorsque la profondeur de l’arbre représentant les règles est non limitée, on peut diviser les classes en se basant sur les caractéristiques jusqu’à arriver à des ensembles homogènes **(0.25)**
* Le temps d’entraînement *T(NB, CART, REG) = (0.0017, 0.0006, 0.0073)*. Expliquer comment chaque algorithme fonctionne en motivant ces durées **(1.5pt)**
  + Naïve Bayes multinomial compte pour chaque classe « c » le nombre des occurrences de chaque valeur « v » d’une caractéristique « f » pour calculer la probabilité conditionnelle P(f=v|c). Ces probabilités sont utilisées pour estimer la probabilité P(c|f). La durée de l’entraînement est dépendante du nombre des classes, le nombre des caractéristiques et le nombre des différentes valeurs dans chaque caractéristique **(0.5)**
  + CART essaye de créer un arbre binaire (SI-SINON) en cherchant la caractéristique qui peut diviser un ensemble de classes en deux ensembles plus homogènes dans chaque nœud. Le critère d’arrêt est lorsqu’on atteint une feuille avec des échantillons de la même classe, la profondeur maximale ou le nombre minimum des échantillons dans une feuille **(0.5)**
  + La régression logistique voit le classement comme un problème d’optimisation ; elle essaye d’optimiser une fonction de coût. Elle représente chaque classe comme une probabilité calculée en combinant les caractéristiques linéairement et en transformant cette combinaison en probabilité avec la fonction logistique. Elle applique des itérations pour optimiser la fonction du coût. A chaque itération, on estime la classe, on calcule l’erreur de classement et on met à jours les poids de combinaison en se basant sur l’erreur. Le temps dépend sur le nombre des échantillons, nombre des caractéristiques (l’estimation de la classe, la mise à jour des paramètres, le nombre des classes (le calcul de l’erreur). **(0.5)**
* Nous avons utilisé la régularisation L2 dans la régression logistique. Expliquer l’intérêt de cette régularisation en indiquant son équation **(0.5)**
  + L’intérêt de la régularisation est d’éviter le sur-apprentissage. Ceci est possible en ajoutant la formule à la fonction du coût pour forcer les thétas à avoir des petites valeurs en plus de minimiser l’erreur. Lambda est un paramètre de régularisation, m est le nombre d’échantillons et n est le nombre des caractéristiques.

**5- Test**

5.a- Prédire les classes de test pour chaque modèle et afficher le rapport de classement. **(0.5pt)**

|  |
| --- |
| ***print("Naive bayes")***  ***print(classification\_report(Y\_test, naive\_bayes.predict(X\_test)))***  ***print("Arbres de decision")***  ***print(classification\_report(Y\_test, cart.predict(X\_test)))***  ***print("Regression logistique")***  ***print(classification\_report(Y\_test, reg.predict(X\_test)))*** |

5.b- Que signifient les résultats suivants :

* F1-score de la classe « Bird » est 1.0 dans les trois modèles **(0.25pt)** Cette classe est bien séparable vu les caractéristiques données
* F1-score de la classe « Reptile » est 0.0 dans les trois modèles **(0.25pt)** Cette classe est non séparable vu les caractéristiques données (les caractéristiques ne sont pas suffisantes)
* La précision d’une classe dans un des modèles est 0.33 et le rappel est 1.0 **(0.25pt)** Le modèle peut détecter tous les échantillons de cette classe, mais il insère plusieurs autres échantillons dans cette classe
* La précision dans un modèle en terme de micro-avg est 0.75 et en terme de macro-avg est 0.62 **(0.25pt)** La micro-avg calcule la performance en prenant en considération toutes les classes à la fois. La macro-avg calcule la performance pour chaque classe d’une façon binaire et fait la moyenne des performances de toutes les classes. Dans notre cas, si nous avons 3 échantillons justes et une fausse, P\_macro-avg = 3/4 ; P\_micro-avg = (1/2 + 1 + 1)/4 = 0.625