PROJET EN AUTONOMIE :

PREDICTIONS AUTOUR DE LA NOUVELLE TESLA Y

Lisez bien tout ce qui va suivre, cela ne sera pas du temps de perdu… Un mauvais calcul : démarrer trop vite en lisant en diagonale…

---

1 Introduction

Le but de ce projet va être de réaliser, pour une société de conseil des prédictions de l'achat ou non de la nouvelle Tesla Y. Vous avez à disposition une base de 195 clients avec 13 variables prédictives et une variable à prédire : la personne a-t-elle acheté cette voiture ou non ? Jetez un œil au fichier **train.txt** : var1 à var13 sont les variables servant à prédire la dernière variable c'est-à-dire la 14ème variable qui est dénommée var14.

Vous devrez donc mettre au point la « moulinette » qui vous semble la meilleure (explications complémentaires à suivre). Une fois ce choix fait, vous faites calculer les prédictions issues de votre meilleure moulinette et ce, en aveugle donc, sur 65 nouveaux individus qui se trouvent dans le fichier **test_EVALUATION.txt.** De notre côté, nous comparerons vos prédictions avec les prédictions qu'il faudrait trouver…

Le délivrable à remettre sur le campus est un fichier .ZIP qui contiendra ceci :

- vos programmes Python
- le fichier « **fichier_mes_predictions_plus_verification_rapide.py** » : où vous aurez remplacé entre la ligne 1 et la ligne 65, les 65 valeurs «bidon» par vos propres prédictions. Vous ne touchez à rien d'autre, par exemple, ne changez pas le nom de la variable : **predictions_65_patients**.
- !!! Vous pourrez vérifier que ce fichier ne comporte pas d'erreur(s) de syntaxe en lançant votre programme **fichier_mes_predictions_plus_verification_rapide.py**
- le fichier « **synthèse_travail_réalisé.odt** où vous allez remplacer les **???** par ce qu'il faut.
- Eventuellement, le fichier Python que vous auriez réalisé dans la partie du bonus (explications à suivre).

2 Quelques conseils

2.1 Pour les réseaux de neurones

Vous allez faire des choses très similaires au bureau d'études. Par exemple, vous pourriez découper votre base de 195 clients comme ceci :

- environ 3/4 d'individus pour la base d'apprentissage (y compris votre ensemble pour le early stopping si nécessaire).
- environ ¼ pour les tests.

Chose importante !!! :

- vous allez tester avec 1 neurone, vous mesurez les performances sur l'ensemble de test
- vous allez tester avec 2 neurones, vous mesurez les performances sur l'ensemble de test
- …
- …

Bien entendu, le meilleur nombre de neurones que vous allez retenir sera celui qui fournit les meilleures performances dans l'ensemble de **test**.

2.2 Pour les arbres de décision

Pas d'early-stopping, donc on pourrait répartir les ensembles comme ceci :

- environ 3/4 d'individus pour la base d'apprentissage
- le restant pour les tests.

2.3 Pour les SVMs

Pas d'early-stopping, donc on pourrait répartir les ensembles comme ceci :

- environ 3/4 d'individus pour la base d'apprentissage
- le restant pour les tests.

3 Le brassage

Pensez à brasser vos données…

4 La base

Les données des 195 clients contiennent 13 variables + une 14ème qui vaut 0 ou 1 selon que le client achète ou non.

Pour lire ce fichier, vous faites par exemple :

```
import pandas as pd

donnees_data_frame =  pd.read_csv ("train.txt" , delimiter=" ")

donnees_ensemble_total= donnees_data_frame.values

print (donnees_ensemble_total)
```

Il faut éventuellement normaliser…

5 Vers la fin du projet

La structure grossière de votre programme sera donc la suivante :

1°) lire la base **train.txt**

2°) réaliser les apprentissages/prédictions

3°) choisir la meilleure moulinette

4°) lire la base **test_EVALUATION.txt La dernière colonne vaut systématiquement 0 (c'est normal, on vous a ainsi caché les vrais résultats…).**

5°) réaliser les prédictions sur ces 65 individus

6 Pour ceux qui s'ennuient – bonus (pour dépasser le 20/20…)

Si vous avez fini trop tôt, vous pouvez peut-être réfléchir à comment essayer d'améliorer les performances.

Par exemple, une technique assez élégante consiste à faire du vote majoritaire, ainsi, une fois qu'on a mis au point les 3 « moulinettes » : réseaux de neurones, arbres de décision et SVM comme vous venez de le faire ; ensuite, on crée une « méta-machine » d'apprentissage en procédant tout simplement par vote :

- pour un nouvel individu : calculer la prédiction du réseau de neurones, calculer la prédiction de l'arbre de décision et enfin calculer la prédiction délivrée par le SVM.

  Le résultat de la prédiction sera simplement un vote majoritaire. Exemple, le réseau de neurones prédit 1, l'arbre de décision prédit également 1, le SVM prédit 0. Cela fait donc 2 « 1 » contre 1 « 0 » donc le « 1 » l'emporte ; donc le résultat de la prédiction, pour cet individu, sera « 1 ».

Il y a d'autres manières, solutions…

7 Pour finir

Tout à la fin de ce TP, relisez ceci :

- Vérifiez bien le contenu de votre fichier :
  « **fichier_mes_predictions_plus_verification_rapide.py** » en le lançant :
  ce programme vérifie entre autres que vous avez bien 65 mesures. Le fichier « **fichier_mes_predictions_plus_verification_rapide.py** » sera peut-être à compléter à la main… c'est ce fichier qui va être noté de manière automatique !!!
  Donc apportez-y un minimum de soin pour faciliter la vie du correcteur ☺

Donc pas de programme qui génère un vecteur final ou que sais-je, il faut que ce fichier contienne les 65 valeurs… !!!

- Vérifiez que l'archive **.ZIP** contient bien en plus :
    - **synthèse_travail_réalisé.odt**
    - vos fichiers Python
    - éventuellement, le fichier Python que vous auriez réalisé dans la partie du bonus.

!! Même si vous avez fait cette dernière partie, vous serez évalués sur le fichier « **fichier_mes_predictions_plus_verification_rapide.py** » par une notation automatique, donc il faudra « se mouiller » et choisir entre les deux fichiers de prédictions pour la remise finale, et oui, le client veut un seul produit final…
