MINI CHALLENGE DE DONNEES

HUA Vi Quang
Collab : huavi

OLIVIER GOUDET
FABIEN GARREAU

UNIVERSITE ANGERS

2020

Présentation du challenge

Lors de ces deux dernières séances de TP, vous allez participer à un mini challenge de classification binaire. Les données qui sont fournies sont des données réelles relatives à une campagne de promotion de prêts bancaires réalisée pour une institution bancaire. Cette campagne marketing s'est déroulée sous la forme d'appels téléphoniques auprès d'un grand nombre de clients, de façon à savoir s'ils étaient prêts à souscrire à un produit bancaire (dépôt à terme de type assurance vie). Pour chaque client interrogé au cours de cette étude, on sait s'il a finalement souscrit un prêt bancaire (réponse oui, classe 1) ou bien s'il n'a pas souscrit (réponse non, classe 0).

A partir des données relevées à propos des clients au cours de cette campagne marketing, le but de ce TP est de prédire si chaque client aura tendance à souscrire au produit bancaire ou non. Ce genre de modèle prédictif pourrait typiquement être utilisé par des banques ou des compagnies d'assurance pour mieux cibler des offres de produit à des clients.

Attributs Les attributs relatifs à chaque client qui ont été collectés lors de cette campagne marketing sont décrits en anglais dans le jeu de données. Ce sont des données mixtes (catégorielles et continues). Il y a des attributs propres aux caractéristiques du client (1-7), des données relatives aux derniers contacts obtenus lors de la campagne d'appel téléphonique (8-14) et enfin des attributs qui définissent le contexte socio-économique général au moment de l'appel (15-19). Les définitions de ces attributs sont les suivantes :

1. ***âge*** : âge du client

2. ***job*** : type d'emploi (ouvrier, employé, chômeur, ...)

3. ***marital*** : statut marital (célibataire, marié, ...)

4. ***education*** : niveau d'éducation

5. ***default*** : a déjà un crédit en défaut de paiement (oui ou non).

6. ***housing*** : a déjà un crédit immobilier (oui ou non).

7. ***loan*** : a déjà un crédit personnel (oui ou non)

8. ***contact*** : type de communication (sur fixe ou portable).

9. ***month*** : mois du dernier contact

10. ***day_of_week*** : jours de la semaine lors du contact.

11. ***duration*** : durée du dernier contact téléphonique.

12. ***campaign*** : nombre de contacts téléphoniques pour ce client durant cette campagne.

13. ***pdays*** : nombre de jours passés depuis la dernière campagne d'appel téléphonique.

14. ***previous*** : nombre de d'appels déjà effectués avant cette campagne et pour ce client.

15. ***poutcome*** : résultat de la dernière campagne de marketing pour ce client (succès, non existant, échec).

16. ***emp.var.rate*** : variation du taux de chômage (indicateur trimestriel).

17. ***cons.price.idx*** : indice du prix à la consommation (indicateur mensuel).

18. ***cons.conf.idx*** : indice de confiance des ménages.

19. ***euribor3m*** : taux interbancaire à trois mois (indicateur journalier).

20. ***nr. employed*** : nombre d'employés.


K plus proches voisin
Regression Logistique
Réseau de neurones

K PLUS PROCHES VOISINS

Onglet 1

DANS CETTE SECTION :

- **Données**
- **Méthode**
- **Résultat**

L'idée de cette méthode c'est pour estimer la classe correspondante à appartenir dans X_train à partir les k plus proches voisin pour chaque nouvelle entrée X des données.

Pour chaque nouvelles entrées X des données, il faut calculer les plus proches voisins dans X_train.

Parmi ces K plus proches voisin de X_train on les Y_label (par example 0 1 0 1 0. Pour k = 5).

La prédiction de la classe à partir ce résultat. (Ici la classe est 0 car il 3 zeros et 2 uns).

Tout d'abord on va charger les données dans un data-Frame avec la fonction de pandas (*pandas. Readcsv*) et un séparateur ','.

Il y 2 données à charger :

- Le 1er ce sont les informations d'une personne comme : "age","job","marital","education","default","balance","housing","loan","contact","day", "month","duration","campaign","pdays","previous","poutcome ».

- Le 2eme c'est juste une colonne 'has suscribed ' avec les valeurs 0 ou 1.

On va définir la fonction ***euclidian_distance*** qui va calculer la distance entre 2 vecteurs

Tester cette fonction avec 2 vecteurs. Le 1er c'est le 1er individu et le 2eme c'est un individu choisi au hasard dans le data-frame.

On va ensuite définir la fonction ***Neighbors*** qui prends un data, un label, un data-test et k valeurs en paramètres et qui renvoie les k voisins les plus proche de data-test.

La fonction ***prédiction*** qui prend les résultats de la fonction Neighbors comme paramètres et qui renvoie dans quelle catégorie que le data-test appartient.

La division le data-frame *en 70% pour le data entrainement* et *30% pour le data test.*

La fonction *taux_error* pour calculer le taux d'erreur en comparant les valeurs test avec les vraies valeurs de label.

La phase test :

J'ai obtenu un score de 0.4840319361

Les résultats sont sauvegardés dans le ficher 'data2.csv '

REGRESSION LOGISTIQUE

Onglet 2

DANS CETTE SECTION :

- **Données**
- **Méthodes**
- **Résultats**

L'idée de cette méthode c'est pour chaque nouvelle X rentrée des données, on doit le passer dans les fonctions linéaires pour pouvoir obtenir un vecteur de sorite de taille K. Ensuite on va faire les prédictions sur ces résultats.

Charger les données dans un data-Frame avec la fonction de pandas (*pandas. Readcsv*) et un séparateur ',' .

Il y 2 données à charger :

- Le 1er ce sont les informations d'une personne comme : "age","job","marital","education","default","balance","housing","loan","contact","day", "month","duration","campaign","pdays","previous","poutcome ».

- Le 2eme c'est juste une colonne 'has suscribed ' avec les valeurs 0 ou 1.

On doit transformer les données sous forme un numpy array avec ***pandas.values***.

Extraire X et Y*. Y est la dernière colonne et X est le reste d'array.*

Créer un vecteur *weights* de taille *(k+1,1). Ce vecteur est pour calculer le produit scalaire entre ce vecteur et chaque entre de X données.*

La fonction ***output*** qui renvoie un vecteur de sortie (on faire des fonctions linéaires dans cette fonctions).

La fonction ***prédiction*** calcule la prédiction à partie des sorties du modèles.

La fonction ***error_rate*** qui renvoie le taux d'erreur en comparent en y_label_prediction et les vrais y_label.

Diviser les données en 70% pour apprentissage et 30% pour le test

La phase test :

Tout d'abord on charge les donées sous forme d'un numpy d'array. On doit ajouter une colonne de 1 car son shape est (1002,51) pour pouvoir faire un produit scalaire avec le vecteur weights.

Ensuite on calcule la prédiction sur la sortie du produit scalaire entre les nouvelles valeurs X rentrées et le vecteur weights.

J'ai obtenu un score de 0.9191616766.

Les résultats sont sauvegardés dans le ficher 'data3.csv '

RESEAU DE NEURONS

DANS CETTE SECTION :

- **Données**
- **Méthode**
- **Résultats**

Un réseau neurone est un ensemble de modèles de régressions logistiques organisés en couches, est aussi appelé perception multi-couche
A travers cette méthode on va voir que le modèle peut apprendre à extraire de l'information de plus en plus haut niveau au fil des couches successives à partir d'informations de très bas niveau

Charger les données dans un data-Frame avec la fonction de pandas (*pandas. Readcsv*) et un séparateur ','.

Il y 2 données à charger :

- Le 1er ce sont les informations d'une personne comme :
  "age","job","marital","education","default","balance","housing","loan","contact","day", "month","duration","campaign","pdays","previous","poutcome ».

- Le 2eme c'est juste une colonne 'has suscribed ' avec les valeurs 0 ou 1.

On doit transformer les données sous forme un numpy array avec *pandas.values.*

On divise les données en 70% pour apprentissage et 30% pour le test.

La fonction *prediction* qui calcule les prédictions (0 ou 1) à partir des sorties du modèle

La function ***error_taux*** qui calcule le taux d'erreur en comparant les y prédits avec les y réels.

Ensuite on doit créer un modèle de régressions logistique multivarié avec class ***Neural_network_binary_classif (th.nn.Module).***

On précise bien le matériel utilisé est ***device = 'cpu'*** et chargement du modèle sur le matériel choisi.

Il faut préciser taux d'apprentissage, définir le critère de Loss et optimiser.

Nb_epoches est le nombre de valeurs de calculer.

La boucle for pour traiter et optimiser les calculs.

La phase test *:*

Sur les résultats obtenus. On obtient les résultats beaucoup plus optimisés. La valeur soit 1 ou 0.

J'ai obtenu un score de 0.8253493014

Les résultats sont sauvegardés dans le ficher 'data.csv '