TP noté "Machine Learning 1"

11 janvier 2023

Les données proposées dans ce sujet sont des données synthétiques qui nous permettront d'examiner et mettre en oeuvre plusieurs des méthodes rencontrées dans l'unité. Ce sujet traitera ainsi de

- régression de Poisson et régression linéaire
- clustering (avec KMeans)
- discrimination avec une régression logistique
- estimation des performances par validation croisée
- estimation d'incertitudes par bootstrap

Instructions

- répondre aux questions en utilisant *le langage de votre choix*, soit dans un Notebook Jupyter, soit dans un notebook Rstudio
- vous ferez les calculs et tracerez les figures demandées. Vous insèrerez vos commentaires dans une cellule markdown/texte. ***Il est impératif de commenter ce que vous faites, pourquoi, et les résultats que vous obtenez ; le code seul ne vous rapportera qu'un peu plus de la moitié des points.***
- vous pouvez même utiliser plusieurs langages si vous le souhaitez
- tous documents autorisés
- interdiction de communiquer avec d'autres intelligences, humaines ou artificielles.
- en fin de session, vous rendrez votre notebook en le téléversant sur icampus

Les parties 2, 3, 4, 5 (partiellement) sont indépendantes

1 - Données

- Charger le fichier `Data_2023.csv` .Vous appellerez `df` le dataframe.
- Combien ce fichier contient-il d'exemples, de variables ?
- Quelles peuvent être les variables catégorielles ?

In [ ]:

In [ ]:

- Examiner quelles sont les corrélations entre les différentes variables

In [ ]:

In [ ]:

- Tracer les histogrammes des variables 1, 3, 5. Peut-on soupçonner qu'il y ait des sous ensembles d'exemples avec des comportements différents ?

In [ ]:

2 - Conversion

La variable 2, `v2`, est catégorielle. Convertir cette variable en binaire, sur deux niveaux 0 et 1 ; et mettre à jour le dataframe `df`. Si vous n'arriviez pas à réaliser cette opération, charger le résultat `Data_2023b.csv`

In [ ]:

In [ ]:

3 - Discrimination (régression logistique)

La réponse est la variable `r`. On cherche à prédire cette réponse `r` à partir des variables explicatives `v1` à `v9`. Si le dataframe contient des variables supplémentaires, vous ne les utiliserez pas pour la discrimination.

- Séparer les données en une base d'apprentissage et une base de test, avec un ratio 2/3, 1/3.
- Apprendre une régression logistique (classe `LogisticRegression` en python/sklearn, fonction `glm` avec le paramètre `family=binomial` en R ou en Python avec statsmodels)
- Calculer les scores sur les bases de test et d'apprentissage. Ces scores sont-ils différents, commenter.
- Calculer la matrice de confusion (sur la base de test !). Quel est le taux de faux positifs ? NB - Sous python, vous pouvez utiliser `pd.crosstab` pour calculer cette matrice de confusion
- Apprendre la régression sur la base complète et calculer le score par validation croisée. Comparer le score obtenu par validation croisée à celui obtenu sur la base de test. Quel est l'intérêt de la validation croisée (au moins si les données sont en nombre faible) ?

NB - la variable `v6` présente des valeurs manquantes NA. **Si** cela posait des difficultés à la méthode que vous employez, ce n'est pas obligé, vous pouvez (i) soit supprimer les lignes correspondantes (ii) soit utiliser le fichier `Data_2023c.csv`

In [ ]:

In [ ]:

In [ ]:

4 - Régression linéaire et de Poisson

On est ennuyé avec les valeurs manquantes dans la variable `v6`. On décide d'imputer ces valeurs manquantes. Pour cela, on va tenter de prédire les valeurs manquantes à partir des autres variables.

Sous R, vous pourrez utilier la fonction `lm`, et sous Python, vous pourrez charger l'équivalent par `from LinearRegression_in_Python_like_in_R import lm`. Vous pouvez également utiliser un modèle linéaire généralisé `glm` ; sous Python vous pourrez utiliser la fonction `glm` de statsmodels, et les fonctions de résumé et de diagnostique selon `from GLMRegression_in_Python_like_in_R import GLMsummary, glm_residplot`

- Suivant les outils que vous utilisez, vous pourrez ou pas utiliser la base `df` avec les valeurs manquantes. Le cas échéant, supprimer toutes les lignes sans NA pour la variable `v6`. Sous Python, vous pourrez utiliser `.dropna`, mais également `.notna()` pour accéder aux indices des lignes ne contenant pas de NA, et `.isna()` pour les lignes avec NA.

4.1 - Régression linéaire

- Effectuer une prédiction lineaire de `v6` en fonction des autres variables explicatives (les variables en `v.`, pas `r`)
- Calculer l'erreur quadratique moyenne entre les valeurs exactes et prédites de `v6`. Vous pouvez éventuellement directement accéder à un prédicteur via une méthode `.predict()`, voire même directement aux residus via un attribut `.residuals`
- Au vu des résultats, quelles sont les variables importantes
- Examiner les graphes de diagnostic. Pouvez vous soupçonner une non linarité, pourquoi ?
- Effectuer une nouvelle prédiction en ajoutant la variable `v3**2`. Que devient alors l'erreur quadratique moyenne ? La variable v5 est-elle utile ?
- Si vous ajoutiez `v5**2` plutôt que `v3**2`, quelle seraient l'erreur quadratique moyenne ? Commentaires.

4.2 - Régression de Poisson

- Effectuer une régression de Poisson de `v6` en fonction des autres variables explicatives (les variables en `v.`, pas `r`)
- Reprendre toutes les autres questions précémment traitées avec la régression linéaire (performances, variables à retenir, graphes de diagnostic, non-linéarité...)
- Comparer les résulats obtenus par les deux approches et commenter

4.3 - Imputation

- Rempacer les données manquantes dans `df` en prédisant les valeurs manquantes à l'aide du modèle linéaire. Si vous ne voyez pas comment faire, utiliser `Data_2023c.csv` dans la suite.
- Enfin, calculez une régression logistique pour prédire la variable r et évaluez les performances (sur une base de test ou par validation croisée, comme en 3). Comparez le score au score obtenu précédentemment.

In [ ]:

In [ ]:

Clustering

Comme vous l'avez peut-être observé, on soupçonne que les données puissent être en fait composées de différents *clusters*, qui possèderaient des caractéristiques différentes. L'idées est alors qu'on pourrait -- peut-être, améliorer les performances de discrimination et utilisant des modèles différents sur chacun des clusters.

Au vu des histogrammes, on teste l'hypothèse de 3 clusters.

- Utiliser la méthode kmeans pour définir 3 clusters. Tracer les histogrammes de répartition des labels identifiés par kmeans.
- Ajouter au dataframe une colonne 'km_labels' contenant le label du cluster pour chaque exemple.

NB - Sous Python

```
from sklearn.cluster import KMeans
kmeans = KMeans(...
```

In [ ]:

In [ ]:

In [ ]:

- Séparer votre dataframe en une base d'apprentissage `train` (les 3000 premières lignes) et une base de test `test` (les 1500 suivantes).
- Pour chacun des groupes (vous pouvez filtrer vos données sur la valeur de la colonne 'km_labels')
  - calculer un modèle de régression logistique
  - et les performances correspondantes (score s_i, taux de faux positifs t_i) sur la base de test, pour les exemples du même groupe
  - si n_1, n_2, n_3 sont les nombres d'exemples dans chacun des groupes, s_1, s_2, s_3 les scores dans chaque groupe, alors le score global sera

$$s = \frac{n_1 s_1 + n_2 s_2 + n_3 s_3}{n_1 + n_2 + n_3}$$

    et pour le taux de faux positifs,

$$t = \frac{n_1 t_1 + n_2 t_2 + n_3 t_3}{n_1 + n_2 + n_3}$$

- Comparez les performances aux performances obtenues précédemment et commentez

In [ ]:

In [ ]:

Évaluation de la stabilité par bootstrap

- Ré-échantillonnez l'ensemble `train` et générez le même nombre d'échantillons, puis apprenez vos modèles et calculez le score global comme précédemment sur la base de test *inchangée*.
- Effectuez ceci disons B=200 fois, en stockant les valeurs de scores et taux de faux positifs.
- Tracez les histogrammes des valeurs obtenues
- Donnez les moyennes et intervalles de confiance correspondant.

In [ ]:

In [ ]:

In [ ]: