Apprentissage supervisé - Regression, DecisionTreeRegressor, RandomForestRegressor

December 8, 2020

Objectifs du Brief

- Réaliser un projet individuel de la préparation de données, d'apprentissage automatique à la phase de prédiction.
- Les données utilisées sont issues d'un recensement de 1990 aux États-Unis. La référence de ces données est : **Pace, R. Kelley et Ronald Barry, "Sparse Spatial Autoregressions", Statistics and Probability Letters, Volume 33, Numéro 3, 5 mai 1997, p. 291-297.**
- L'objectif de ce projet d'apprentissage automatique est de pouvoir, à partir de plusieurs caractéristiques (features), d'estimer les prix des maisons (target) dans l'état de la californie.
- L'interprétation des résultats (et non pas juste leur affichage)
- La prise en main des bibliothèques (Pandas, Matplotlib, Scikit-learn, Numpy)

1 Préparation des données

1.1 Téléchargement de données

Téléchargez le jeu de données housing.csv. Placez le dans le même répertoire de votre fichier (Python ou Notebook).

1.2 Information sur les données

1. Créez un code qui lit le fichier "housing.csv" et affiche ses premières lignes. Pour ce faire, utilisez les fonctions **"read_csv"** et "head" de la bibliothèque pandas. Sachant que la valeur cible est **"median_house_value"**, traitons-nous un problème de classification ou de régression ?
2. Créez un code qui affiche le nombre de lignes et de colonnes des données, le type des attributs et le nombre de valeurs non nulles. Quelle remarque sur l'attribut **"total_bedrooms"** par rapport aux autres attributs ?
3. A travers la question précédente, vous avez du remarquez que le type dans valeurs utilisées dans l'attribut **"ocean_proximity"** est un objet (forcément un texte vu qu'on manipule un fichier CSV). Il est intéressant de connaître ses valeurs. Pour cette finalité, créez un code qui affiche l'occurrence des valeurs utilisées dans cet attribut.

By Youssef Mourchid

4. Créez un code qui affiche des statistiques sur les attributs de ton jeu de données.
5. Créez un code qui affiche les histogrammes des différents attributs. Le nombre de "bins" à saisir est 50 et la taille de chaque histogramme "figsize=(20,15)".

1.3 Répartition des données

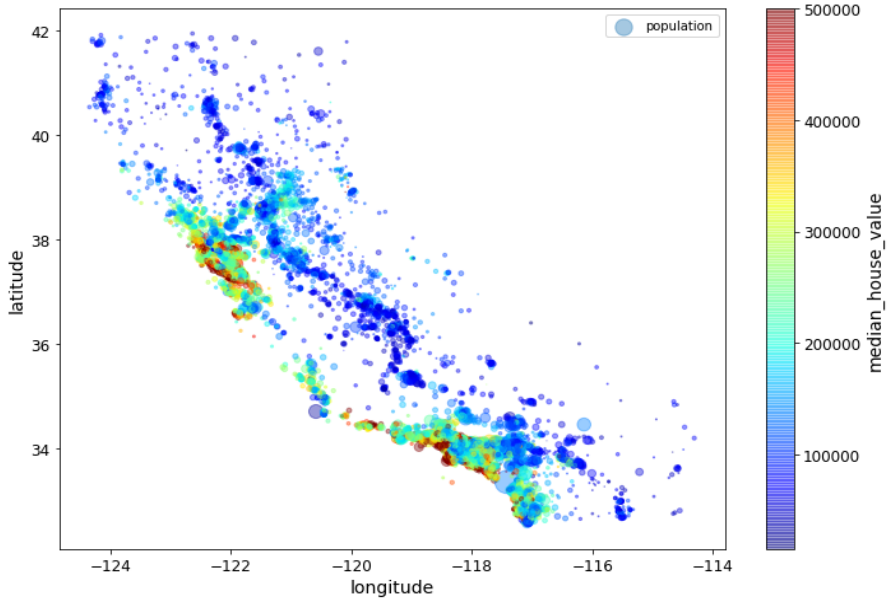
1. Créez un code qui partitionne les données en base d'apprentissage et base de test. Optez pour 80% pour l'apprentissage et 20% pour le test.
2. Affichez l'en-tête de la base de test

Nous nous intéresserons par la suite uniquement à la base d'apprentissage. Pour cette raison, le terme données fera référence à la base d'apprentissage.

1.4 Découverte et visualisation des données

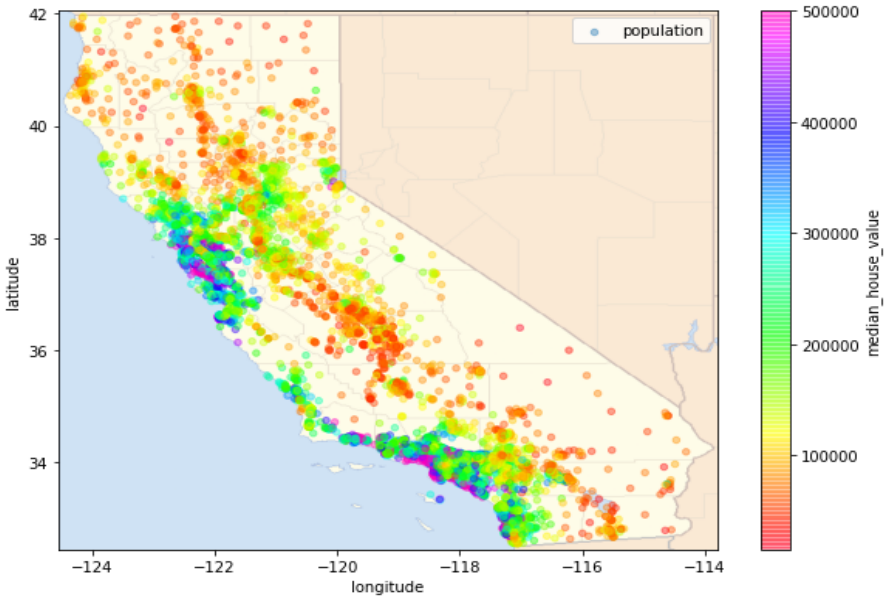
L'information géographique (latitude et longitude) existe dans la base de données, il est intéressant de créer des graphes illustrant une visualisation géographique des données.

1. Créez un code qui affiche en abscisse la longitude et en ordonnée la latitude. Optez pour le type scatter dans la fonction plot pour l'affichage et une valeur d'alpha (c'est un paramètre qui joue sur la transparence de la courbe) de 0.1 pour un affichage plus clair.
2. Créez un code qui permet d'avoir une idée sur le lien entre la position géographique et le prix des maisons (target). Optez pour une valeur égale à False de "sharex".



3. Modifiez le graphe de la question précédente pour savoir la raison du prix élevé de quelques maisons. Pour ce faire :
   - Téléchargez l'image de la californie
   - Utilisez la fonction imread du sous-module image du module matplotlib

- Utilisez la fonction imshow du sous-module pyplot du module matplotlib



4. Une pratique très intéressante dans l'analyse de données est l'étude des corrélation entre les variables. Créez un code qui affiche, en valeur, la corrélation de l'attribut **"median_house_value"** avec les autres attributs. Qu'est ce que vous remarquez ?

1.5 Nettoyage des données

Avant d'intégrer les données dans un algorithme d'apprentissage automatique, il est indispensable de séparer le "features" et la valeur cible (target).

1. Créez un code permettant de créer deux variables :
   - Une première contenant que les input. Utilisez la fonction drop du module pandas
   - Une deuxième contenant que les labels. Utilisez la fonction copy du module pandas
2. Dans la question 8, vous avez dû remarquer que l'attribut **"total_bedrooms"** a des valeurs manquantes (NaN). Pour remédier à ceci, il existes trois options :
   - Supprimer les valeurs manquantes (NaN)
   - Supprimer l'attribut **"total_bedrooms"**
   - Remplacer les valeurs manquantes par une autre valeur (0, la moyenne, la médiane, ...). Nous optons pour cette méthode. Ecrivez un code qui remplace les valeurs manquantes par la médiane. Utilisez les fonctions median et fillna du module Pandas. Vérifiez avec la fonction "info" si le problème a été résolu.
3. Les algorithmes d'apprentissage profond préfère de travailler avec les données numériques. Ceci est valable pour tous les attributs sauf **"ocean_proximity"**. Vérifiez ceci en affichant 10 de ces valeurs. Transformer les valeurs qualitatives en des valeurs numériques.
4. Affichez les données pour vérifier le résultat.

2 Sélection, apprentissage et évaluation du modèle

1. Créez un code permettant d'appliquer la régression linéaire sur les données d'apprentissage.
2. Créez un code qui prédit les classes de la base d'apprentissage. Pour ce faire, utilisez la méthode predict de la classe LinearRegression en donnant comme argument les données d'apprentissage. Ensuite, affichez les valeurs cible réelles et celles prédites.
3. Calculez la mesure RMSE du modèle de la régression linéaire.
4. Refaites les deux étapes précédentes avec le modèle DecisionTreeRegressor. Calculez la mesure RMSE du modèle DecisionTreeRegressor qui existe dans le sous-module tree du module sklearn.

   Pour plus d'informations sur les arbres de décision:

   http://cedric.cnam.fr/vertigo/cours/ml2/tpArbresDecision.html
5. Même si la valeur de RMSE de DecisionTreeRegressor est égale à 0, on ne peut pas conclure que ce modèle fonctionne parfaitement sur la base d'apprentissage. Pour s'assurer, on va répartir la base d'apprentissage en base d'apprentissage et en base de test en utilisant la méthode 10-fold cross-validation. Pour ce faire, utilisez la fonction cross_val_score du sous-module model_selection du module sklearn. Ensuite, affichez :
   - La valeur MSE de chaque fold
   - La moyenne des MSE de tous les folds
   - L'écart type de tous les folds
6. Suivre les étapes de la question précédente sur le modèle de la régression linéaire. Ensuite, comparez les résultats avec ceux du DecisionTreeRegressor. Quel modèle présente un problème d'apprentissage ? pourquoi ?

3 Fine-Tunning

3.1 Grid Search

Dans cette, partie nous allons chercher les paramètres du modèle de régression qui donnent les meilleurs résultats

1. Écrire un code qui :
   - Crée un objet de la classe RandomForestRegressor

     Pour plus d'informations sur RandomForestRegressor:

     https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb
   - Crée la variable suivante :

     $$param\_grid = [' n\_estimators' : [3, 10, 30], ' max\_features' : [2, 4, 6, 8]]$$

     Cette variable contient un dictionnaire avec quelques valeurs de deux paramètres de la méthode RandomForestRegressor. Au total, 4x3=12 combinaisons vont être testées.

- Applique une recherche, de type GridSearch, du couple qui donne le meilleur résultat. Pour ce faire, utilisez la fonction GridSearchCV du sous-module model_selection du module sklearn. Optez pour une valeur de 5 pour "cv" (une validation croisée de type 5-fold cross-validation)

2. Affichez les meilleurs paramètres de la méthode RandomForestRegressor en utilisant la fonction best_params_
3. Affichez les résultats des 12 combinaisons avec la fonction grid-search

3.2 Evaluation sur la base de test

Testez votre modèle d'apprentissage sur la base de test. Pour ce faire, pensez à :

1. Remplacer les valeurs NaN de l'attribut "total_bedrooms" de la base de test par la médiane
2. Transformer les valeurs textuelles de "ocean_proximity" en valeurs numériques
3. Stocker le modèle d'apprentissage dans une variable en utilisant la fonction best_estimator_ du module GridSearchCV
4. Calculer la valeur RMSE du modèle sur la base de test