Intelligence artificielle

- FISA4 -

TP1 : Apprentissage supervisé - Régression -

Khadidja OULD AMER (khadidja.ould-amer@isen-ouest.yncrea.fr)

Objectifs du TP

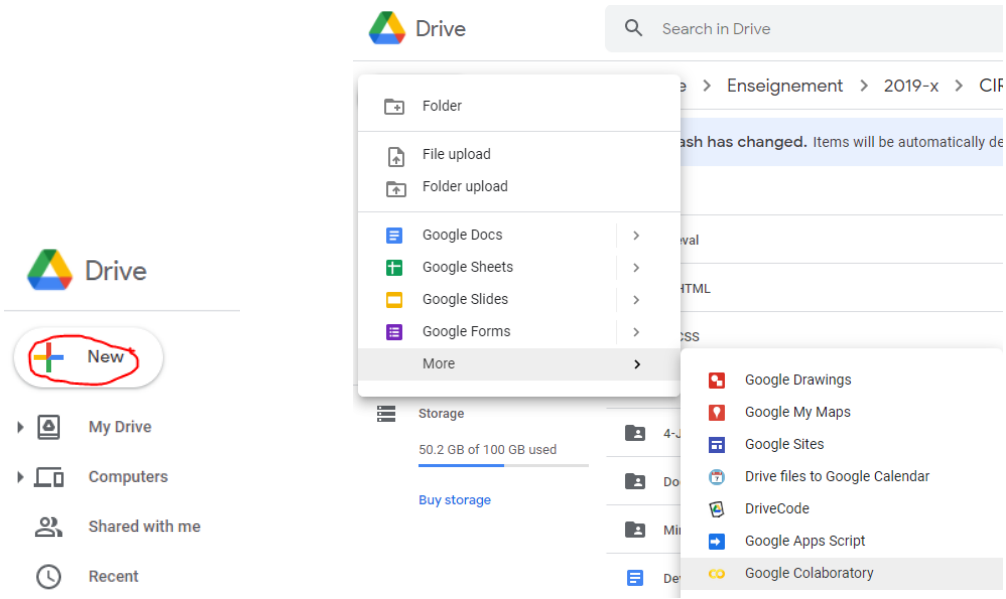
- Assimiler les différentes étapes d'un projet d'apprentissage automatique traitant la régression
- La prise en main des bibliothèques de la partie Liens utiles et savoir utiliser, idéalement, leurs documentations
- **Rédiger un compte rendu et répondre aux questions en fin de TP.**

Liens utiles

- Pandas
- Matplotlib
- Scikit-learn
- Numpy

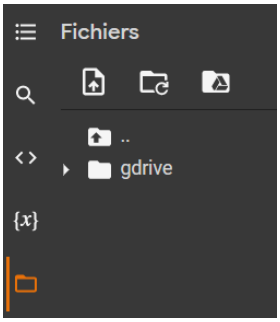
Préparation de l'environnement de travail

1. Depuis votre Google Drive,
    a. créez un dossier "tp_IA" dans lequel vous allez stocker les solutions de tous les TP
    b. dans le dossier "tp_IA", créez un dossier TP1
    c. créez un notebook GoogleColab, nommé tp1_IA dans le dossier TP1



2. Les données utilisées à l'intérieur de GoogleColab sont supprimées automatiquement après la fin de la session. Pour remédier à ceci, il est recommandé de stocker ces données sur GoogleDrive et d'y accéder via GooglColab.
   Installez GoogleDrive dans votre environnement d'exécution :

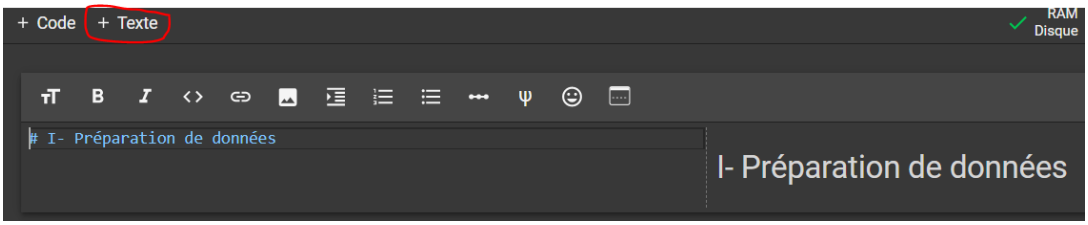
```
from google.colab import drive
drive.mount("/content/gdrive/")
```



Remarque :
vous allez remarquer que mon notebook est en dark mode, si vous voulez l'activer cliquez sur Tools > Settings > Site > Theme > dark

I- Préparation de données

3. Créez une section intitulée I- Préparation de données



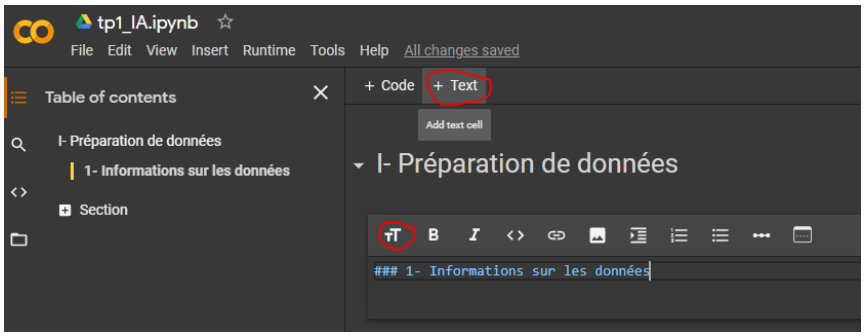
0- Téléchargement de données

4.

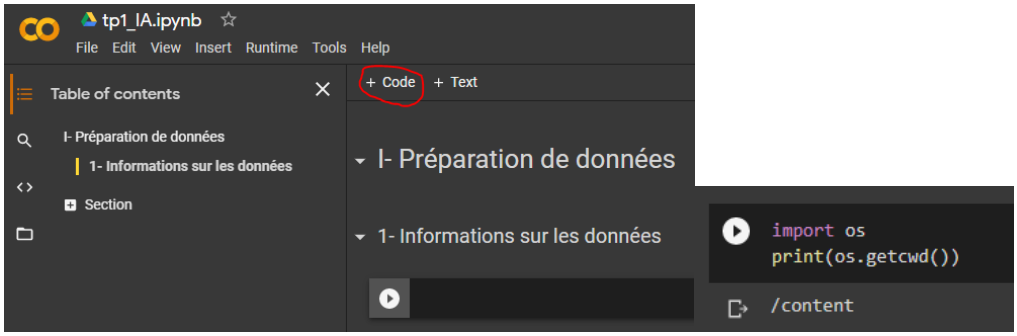
a. Téléchargez housing.csv. Ensuite, déplacez-le dans le dossier TP1 dans GoogleCloud.
   C'est une base de données permettant d'effectuer la régression pour prédire les prix des maisons (= output = valeur cible) dans l'État de la Californie à partir de plusieurs caractéristiques (=input). Il s'agit des données 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.
b. Vérifiez si la base de données existe aussi sur GoogleCollab (dans le dossier "gdrive")

1- Informations sur les données

5. Créez un titre intitulé 1- Informations sur les données (il va être automatiquement inséré comme sous-section de la section I)



6. Créez et exécutez, via le signe de play, le code ci-dessous qui permet de localiser le chemin du travail



7.

a. Créez un code qui lit le fichier "housing.csv" et stocke son contenu dans une variable (de type DataFrame) en utilisant la fonction "read_csv" de la bibliothèque pandas
b. Affichez les longitudes de cette base de données en utilisant le code suivant : print(nom_variable_question7a["longitude"])
c. Affichez les 5 premières lignes de "housing.csv" en utilisant la fonction "head" de la bibliothèque pandas

d. Sachant que la valeur cible à prédire est "median_house_value", il s'agit d'un problème de classification ou de régression ?

8.

a. 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. Pour ce faire, utilisez la fonction "info" de la bibliothèque pandas.
b. Qu'est ce que vous remarquez à propos de l'attribut "total_bedrooms" par rapport aux autres attributs ?
c. Qu'est ce que vous remarquez à propos de l'attribut "ocean_proximity" par rapport aux autres attributs ?

9. Créez un code qui affiche l'occurrence des valeurs utilisées dans l'attribut "ocean_proximity" en utilisant la fonction "value_counts" de la bibliothèque pandas
10. Créez un code qui affiche des statistiques sur les features et la valeur cible. Pour ce faire, utilisez la fonction "describe()" de la bibliothèque pandas
11. Utilisez le code suivant pour afficher les histogrammes des différents features (datahousing est la variable qui contient les données de "housing.csv"):

```
import matplotlib.pyplot as plt
datahousing.hist(bins=50, figsize=(20,15))
plt.show()
```

2- Répartition des données

12. Créez un titre, intitulé 2- Répartition des données (il va être automatiquement inséré comme sous-section de la section I)
13. Créez un code qui partitionne les données en base d'apprentissage et base de test. Pour ce faire, utilisez la fonction "train_test_split" du sous-module "model_selection" du module "sklearn". Utilisez 80% des données pour l'apprentissage et 20% pour le test.
14. Affichez les premières instances de la base de test avec la fonction "head" du module pandas

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. La base de test sera utilisée vers la fin du TP

3- Découverte et visualisation des données

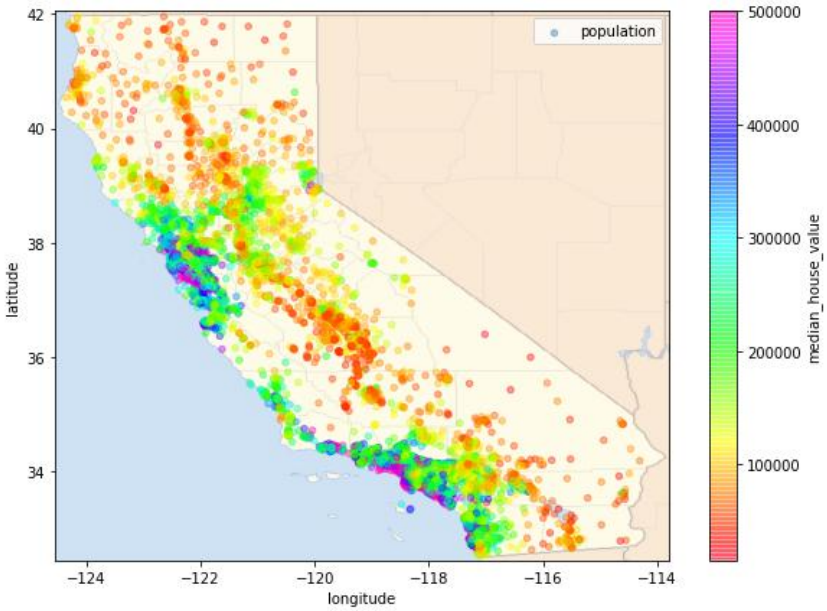
L'information géographique (latitude et longitude) existe dans la base de données, il est donc intéressant de créer des graphes illustrant une visualisation géographique des données.

15. Créez un titre, intitulé 3- Découverte et visualisation des données
16. Téléchargez la carte de la californie et déplacez-la dans le TP1 sur GoogleDrive

17. Exécutez et comprenez le code suivant qui permet d'avoir une idée sur le lien entre la position géographique et le prix des maisons (valeur cible) :

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

california_img = mpimg.imread("/content/Datasets/housing/california.png")
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
label="population", c="median_house_value", colormap='gist_rainbow', colorbar=True,
sharex=False, figsize=(10,7))
plt.imshow(california_img, extent=[-124.55, -113.80, 32.45, 42.05], alpha=0.5,
           cmap=plt.get_cmap("jet"))
plt.legend()
```



18. Une pratique très intéressante dans l'analyse de données est l'étude des corrélations entre les variables. Créez un code qui affiche, en valeur, la corrélation de l'attribut "median_house_value" avec les autres attributs. Pour ce faire, appliquez la fonction "corr" du module Pandas sur les données pour calculer le coefficient de corrélation de Pearson . Qu'est ce que vous remarquez ?

4- Nettoyage des données

Avant d'intégrer les données dans un algorithme d'apprentissage automatique, il est indispensable de séparer les "features" (l'input) et la valeur cible (l'output = valeur à prédire)

19. Créez un titre intitulé 4- Nettoyage des données
20. Créez un code permettant de créer deux variables :

a. Une première contenant les inputs (les features). Utilisez la fonction drop du module pandas pour supprimer la colonne des valeur cible (median_house_value)
b. Une deuxième contenant les valeurs cibles (median_house_value). Pour ce faire , utilisez la fonction copy du module pandas.

21. 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 :

    a. Supprimer les valeurs manquantes (NaN)
    b. Supprimer l'attribut "total_bedrooms"
    c. Remplacer les valeurs manquantes par une autre valeur (0, la moyenne, la médiane, …). Nous optons pour cette méthode. Écrivez un code qui remplace les valeurs manquantes par la médiane de toutes les valeurs de "total_bedrooms". Pour ce faire, utilisez les fonctions "median" et "fillna" du module Pandas. Ajoutez l'argument " inplace=True" à la fonction "fillna". Vérifiez avec la fonction "info" si le problème a été résolu

22. Les algorithmes d'apprentissage automatique préfèrent travailler avec des données numériques. Ceci est valable pour tous les attributs sauf "ocean_proximity". Vérifiez ceci en affichant les valeurs de ce feature
23. Pour transformer les valeurs textuelles en des valeurs numériques, suivez les étapes suivantes :

    a. Créez un objet de la classe OrdinalEncoder de Scikit-Learn. Cette classe est à importer du sous-module preprocessing du module sklearn
    b. Stockez dans une variable les valeurs de l'attribut "ocean_proximity". Utilisez plutôt la syntaxe suivante :
    housing_cat = datahousing[["ocean_proximity"]]
    c. Appliquez la méthode "fit_transform" de la classe OrdinalEncoder sur l'objet créé. L'argument de cette méthode sera les valeurs de l'attribut ocean_proximity. Les valeurs résultantes seront les nouvelles valeurs de l'attribut "ocean_proximity"
    d. Affichez les dix premières instances de la base de données pour vérifier le résultat

II- Sélection et apprentissage du modèle

24. Créez un titre, avec une grande taille de police, intitulé II- Sélection et apprentissage du modèle

1- Apprentissage des données

25. Créez un titre intitulé 1- Apprentissage des données

26. Créez un code permettant d'appliquer la régression linéaire sur les données d'apprentissage. Pour ce faire,

    a. Créez un objet de la classe LinearRegression de Scikit-Learn. Cette classe est à importer du sous-module linear_model du module sklearn
    b. Appliquez la méthode fit sur cet objet en donnant comme arguments les données d'apprentissages et leurs labels (valeurs cibles)

2- Evaluation du modèle d'apprentissage sur les données d'apprentissage

27. Créez un titre intitulé 2- Evaluation du modèle d'apprentissage sur les données d'apprentissage
28. 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 labels réels et ceux prédits
29. Calculez la mesure RMSE du modèle de la régression linéaire. Pour ce faire, utilisez la fonction mean_squared_error du sous-module metrics du module Scikit-Learn. Pensez à appliquer la racine carrée au résultat de la fonction
30. Pour l'apprentissage, utilisez cette fois-ci la classe DecisionTreeRegressor qui représente l'algorithme de l'arbre de décision utilisé pour la régression
31. Calculez la mesure RMSE du modèle DecisionTreeRegressor qui existe dans le sous-module tree du module sklearn

3- Evaluation du modèle d'apprentissage sur les données de validation

Créez un titre intitulé 3- Evaluation du modèle d'apprentissage sur les données de validation

32. Même si la valeur de RMSE de DecisionTreeRegressor est égale à 0, on ne peut pas conclure que ce modèle fonctionne parfaitement. On doit étudier sa capacité de généralisation sur la base de validation.
    Répartissez la base d'apprentissage en base d'apprentissage et en base de validation 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. Affectez la valeur "neg_mean_squared_error" pour l'argument "scoring". Ensuite, affichez :

    a. Le valeur RMSE de chacun des 10 fold. Pensez à ajouter la racine et à multiplier x (-1)
    b. La moyenne des RMSE de tous les folds via la fonction mean()
    c. L'écart type de tous les folds via la fonction std()

33. Appliquez les étapes de la question 32 sur le modèle de la régression linéaire. Ensuite, comparez les résultats avec ceux du DecisionTreeRegressor.

34. Quel modèle présente le problème de sur-apprentissage ? Pourquoi ?

III- Fine-tuning

Créez un titre intitulé III- Fine-tuning

1- Grid Search

Dans cette partie, vous allez chercher les meilleurs paramètres d'une autre méthode de régression à savoir les forêts aléatoires en utilisant la classe RandomForestRegressor

35. Créez un titre intitulé 1- Grid Search
36. Ecrire un code qui :
    a. Crée un objet de la classe RandomForestRegressor. Cette classe existe dans le sous-module ensemble du module sklearn
    b. 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
    c. Applique une recherche, de type GridSearch, pour trouver le couple qui donne le meilleur résultat. Pour ce faire :
        i. créez un objet de la classe GridSearchCV du sous-module model_selection du module sklearn. Le constructeur de cette classe doit avoir :
            1. l'objet de la question 36a
            2. une valeur de 5 pour "cv" (une validation croisée de type 5-fold cross-validation)
            3. une valeur de "scoring" de "neg_mean_squared_error"
        ii. appliquez la méthode "fit" sur l'objet créé dans la question 36.c.i tout en donnant comme argument la base d'apprentissage et ses labels
37. Les meilleurs paramètres obtenus par la méthode GridSearch existe dans 'attribut "best_params_" de l'objet créé dans la question 36.c.i. Affichez-les
38. Affichez les résultats des 12 combinaisons avec le code suivant :

```
pd.DataFrame(grid_search.cv_results_)
```

    Avec grid-search est l'objet créé dans la question 36.c.i

2- Evaluation du modèle d'apprentissage sur les données de test

39. Créez un titre intitulé 2- Evaluation du modèle d'apprentissage sur les données de test
40. Testez votre modèle d'apprentissage sur la base de test. Pour ce faire, pensez à :

    a. Récupérer les données de test et ses labels
    b. Remplacer les valeurs NaN de l'attribut "total_bedrooms" de la base de test par la médiane
    c. Transformer les valeurs textuelles de "ocean_proximity" en valeurs numériques
    d. Stocker le modèle d'apprentissage dans une variable en utilisant l'attribut best_estimator_ sur l'objet créé dans la question 36.c.i
    e. Calculer la valeur RMSE du modèle sur la base de test

Questions – TP1

Préparation des données

1. Quelle est la nature du problème étudié (classification ou régression) et pourquoi ?
2. Que signifie la présence de valeurs manquantes dans un jeu de données ? Quelles sont les méthodes possibles pour y remédier ?
3. Pourquoi est-il important de séparer les données en un jeu d'apprentissage et un jeu de test ?
4. Pourquoi est-il nécessaire de transformer les variables catégorielles en variables numériques avant l'apprentissage ?
5. Quelle est la différence entre supprimer les valeurs manquantes et les remplacer par une statistique (moyenne, médiane, etc.) ?
6. Quels risques pourrait-on rencontrer si on ne traite pas correctement les valeurs manquantes ou les variables catégorielles ?

Sélection et apprentissage du modèle

11. Expliquer le principe de fonctionnement de la régression linéaire.
12. Expliquer le principe de fonctionnement d'un arbre de décision pour la régression (DecisionTreeRegressor).
13. Quelle métrique est utilisée dans ce TP pour évaluer les modèles ? Pourquoi est-elle pertinente dans un problème de régression ?
14. Pourquoi le DecisionTreeRegressor peut-il donner un RMSE égal à 0 sur les données d'apprentissage ? Que signifie ce résultat ?
15. Quelle est la différence entre une évaluation sur les données d'apprentissage et sur les données de validation ?
16. Quel modèle présente des signes de sur-apprentissage ? Justifiez.
17. À quoi sert la validation croisée (cross-validation) dans l'évaluation des modèles ? 18. Pourquoi utilise-t-on un jeu de test séparé après l'étape de fine-tuning ?