Intelligence Artificielle (I-ILIA-028) -- Travaux Pratiques

TP 2 : Machine Learning pour la prédiction de prix de maisons

Après avoir téléchargé, analysé, prétraité et visualisé les données, nous allons nous concentrer durant ce TP sur le développement de modèles *Machine Learning* tel que la régression linéaire, arbres de décisions, les forêts aléatoires, etc. Ces modèles seront utilisés pour la prédiction de prix de maisons de l'état de Californie. Ce TP2 est composé de 3 parties

1. **Partie 1 :** Prise en main avec le processus d'apprentissage et le framework Pytorch
2. **Partie 2 :** Développement et évaluation de modèles *Machine Learning*
3. **Partie 3 :** Analyse comparative à large échelle entre modèles Machine Learning avec Pycaret

- **Partie 1 : Prise en main avec le processus d'apprentissage et Pytorch :**

Afin de mieux comprendre la procédure d'apprentissage avec le framework Pytorch, nous vous proposer de commencer avec un simple exemple ***(f(x) = ax + b)*** avant de progresser plus tard vers des modèles Machine Learning puis de modèles Deep Learning présentant une grande complexité architecturale.

Dans cet exemple, la base de données est un ensemble de **1000 paires (X, Y)** :

- **X** : variable d'entré (***input***) ;
- Y : variable cible (***Target***), ou ***Y = a*X + b*** (**a= 2** et **b = 3** dans notre cas).

Nous proposons de travailler un simple réseau de neurone (une couche entrée et une couche sortie) permettant d'approximer la fonction ***Y = a * X = b***. En d'autres termes, l'objectif est de trouver **a** et **b**.

- **Instruction 0 :** télécharger et compléter le code de démarrage « **Input_TP2_IA.ipynb** »
- **Instruction 1** : importer les librairies nécessaires
- **Question 1 :** créer des données synthétiques et afficher la courbe des paires de valeurs (X, Y)
- **Question 2 :** convertir les données en tensors sous pytorch en utilisant **torch.FloatTensor**.

Note : Les tensors : calcul multidimensionnel, adaptés à un calcul parallèle et accéléré avec GPU

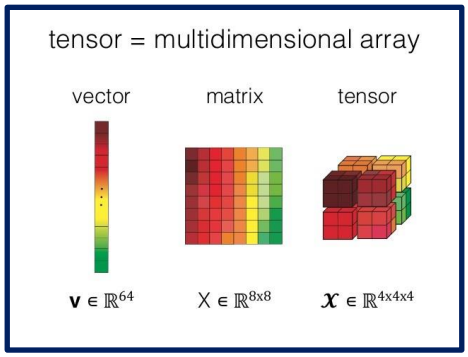

Figure 1: exemple de calcul de tensors

- **Instruction 2** : définir le modèle neuronal suivant en utilisant **nn.Linear.**

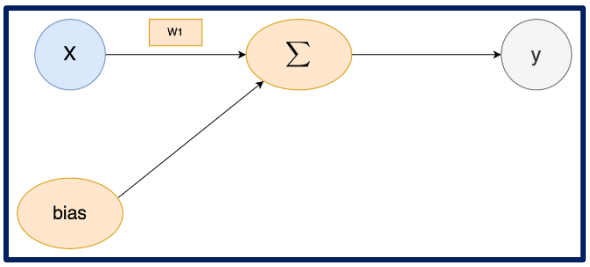

Figure 2: première architecture neuronale (simple)

Tels que W1 et le biais seront calculé par le modèle durant l'apprentissage

- **Question 3** : initialiser le modèle, la fonction de perte et l'optimiseur (descente de gradient avec un pas d'apprentissage de 0,01)
- **Question 4** : définir la boucle d'entrainement et entrainer votre modèle
- **Question 5** : afficher les paramètres (poids) appris par votre modèle.

• Partie 2 : Développement et évaluation de modèles Machine Learning :

- **Instruction 3 :** importer les librairies nécessaires
- **Instruction 4 :** télécharger et prétraiter les données de prix de maisons (TP1)
- **Question 5** : normaliser les données en utilisant **MinMaxScaler**.

Note :

- Vous aurez besoin de redimensionner la variable cible comme suit : **y.values.reshape(-1,1)**
- Utiliser **fit_transform** sur la base d'entrainement et **transform** sur la base de test

- **Instruction 6 :** définir les fonctions de calcul des métriques d'évaluation de modèles

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Sachant que ***At*** est la variable cible et ***F_t*** est la prédiction. **n** est le nombre total d'observations

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Sachant que $\mathbf{y_i}$ est la variable cible, et $\hat{\mathbf{y}}$ la valeur de prédiction.

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$$

Sachant que **yi** est la variable cible, et **xi** la valeur de prédiction.

- **Question 6** : créer et entraîner un modèle de **régression linéaire** avec vos données.
- **Instruction 7** : évaluer votre modèle avec les données d'entraînement et de test.
- **Instruction 8** : afficher le plot des prix réels vs. prix prédits.

- **Question 7 :** créer et entraîner un modèle « ***arbre de décision*** » avec vos données.
- **Question 8** : évaluer ce modèle avec les données d'entraînement et de test.
- **Question 9** : afficher le plot des prix réels vs. prix prédits.
- **Question 10 :** créer et entraîner un modèle « ***arbre de décision*** » avec ces paramètres :
  - Profondeur maximale de l'arbre (***Max Depth***) : 10
  - Nombre minimum d'échantillons dans une feuille (***Min Samples Leaf***) : 4
- **Question 11** : évaluer ce modèle avec les données d'entraînement et de test.
- **Question 12** : afficher le plot des prix réels vs. prix prédits.

  Note : lisez la documentation pour plus d'informations.

- **Question 13 :** créer et entraîner un modèle « ***Random Forest*** » avec vos données.
- **Question 14** : évaluer ce modèle avec les données d'entraînement et de test.
- **Question 15** : afficher le plot des prix réels vs. prix prédits.

Partie 3 : Analyse à large échelle de modèles Machine Learning avec Pycaret

PyCaret est une bibliothèque Python simple et facile à utiliser pour tester/comparer les modèles Machine Learning (non Deep Learning). Pour l'utiliser, nous vous proposons de suivre les étapes suivantes :

- **Instruction 11 :** installer Pycaret et importer les librairies nécessaires
- **Instruction 12** : récupérer et préparer les mêmes données prétraitées précédemment
- **Instruction 13 :** utiliser la fonction « ***setup*** **»** spécifiant les données d'entraînement (X_i) et cibles (Y).
- **Instruction 14 :** utiliser la fonction « ***compare_model*** **»** comparant les modèles Machine Learning
- **Instruction 15 :** optimiser les paramètres du meilleur modèle à l'aide de la fonction « ***tune_model*** »
- **Instruction 16** : évaluer le meilleur modèle ***tuned_model*** avec la fonction « ***evaluate_model*** **»**
- **Instruction 17 :** analyser le graphe « ***feature importance*** » et en tirez vos conclusions.
- **Instruction 18** : utiliser la fonction **predict_model** pour comparer prix réels et prix prédits.
- **Question 16** : analyser l'influence du mélange des données sur le résultat du meilleur modèle présenter votre analyse complète du problème de prix de maisons avec modèles ***Machine Learning***.