Introduction à l'IA

Partie 1 – Préparation de données et Machine Learning

Angelo.Steffenel@univ-reims.fr

Partie 1 – Environnement

Vous pouvez exécuter cet exercice à l'aide de Google Colab (ou d'un jupyter notebook, si vous l'avez installé sur votre ordinateur).

Partie 2 – Récupération du dataset avec Pandas

Vous devez charger un dataset sur les passagers du Titanic (*déjà vu ?*)
Pour cela, vous allez importer la bibliothèque pandas avec l'alias **pd** et charger un DataFrame **titanic** à partir d'un fichier csv. Attention, ce fichier a un séparateur ";" à renseigner :

```
import pandas as pd
# la commande suivante sert à afficher toutes les colonnes (pas de …)
pd.options.display.max_columns=100

titanic = pd.read_csv("https://bit.ly/2C2Tup2", sep=';')
```

Votre première tâche est celle d'utiliser ***.head()*** et ***.tail()*** pour afficher la première et la dernière entrée de ce dataset. Obtenez aussi des informations avec ***.info()*** et ***.describe()***

Partie 3 – Types de données

Grâce à ***.info()*** , vous observez que certaines colonnes sont reconnues avec le type "Object", alors qu'on pourrait les transformer en "category". Modifiez le type de donnée des colonnes 'sex', 'pclass', 'survived' et 'embarked' en utilisant ***.astype("category")***.

Si vous voulez connaître combien d'individus sont recensés dans chaque catégorie, vous pouvez utiliser la fonction ***.value_counts()*** appliquée à chaque colonne.

Partie 4 – Données manquantes et hors-format

Si vous avez fait attention à la sortie de ***.info()***, la colonne 'age' manque certaines cases (des valeurs NaN sont utilisés à leur place). De plus, certains âges sont représentés en tant que valeurs réelles (0,85 par exemple), mais utilisant une virgule au lieu d'un point (ce qui est attendu par l'ordinateur).

Utiliser, à tour de rôle, des fonctions `str.replace()`, `fillna(28)` et `astype("float32")` pour remplacer la virgule, remplir les cases vides avec la valeur "28" et finalement changer le type de données en float32.

Étudiez la sortie de **.info()** et **.describe()** après ces modifications.

Partie 5 – supprimer les colonnes "inutiles"

Certaines colonnes ne seront pas utilisées dans notre exercice, vous pouvez les supprimer avec la fonction ***.drop()***. On donnera comme paramètres un array avec les noms des colonnes à supprimer ('body','ticket','fare','cabin','name','boat','home.dest'), ainsi que l'option "axis=1" pour indiquer qu'on souhaite supprimer les colonnes.

Partie 6 – Graphique "camembert"

On veut afficher sur un graphique "camembert" les distributions des colonnes 'survived' et 'pclass'. Pour cela, on utilisera la bibliothèque Matplotlib.

Si l'image ne s'affiche pas sur l'écran, vous pouvez enregistrer l'image avec plt.savefig('camembert.png') et le visualiser en cliquant dessus.

Partie 7 - BoxPlot

Maintenant, on fera un graphique "boxplot" à partir de la colonne "age". Bien qu'il soit possible de le faire avec Matplotlib, il est bien plus simple (et joli) de le faire en utilisant la bibliothèque Seaborn.

Ensuite, faites un autre boxplot avec deux colonnes cible : pclass et age

Partie 8 - Histogrammes

Toujours intéressés par la colonne "age", nous voulons maintenant faire un histogramme. Avec Matplotlib, on peut indiquer le nombre de « segments », essayez avec bins=8.

Partie 9 - Pairplots

Parfois on cherche des corrélations entre les variables. Les pairplots sont des importantes aides visuelles, croisant différentes variables dans un même écran.
Faire un pairplot avec Seaborn, en utilisant les colonnes `'pclass','sex','age','embarked' et 'survived'.`

Partie 10 – Données catégorisées

Au lieu d'afficher les âges, nous voulons les regrouper en 4 groupes : enfants, jeunes, adultes et seniors. Utiliser la fonction ***.cut()*** pour générer une nouvelle colonne avec ces catégories. Vous pouvez faire un découpage simple ou indiquer les rangées d'âge pour chaque classe.

Faire un graphique camembert avec la répartition des classes.

Partie 11 – Machine Learning pt 1 (partie « challenge »)

On a fini la préparation des données, maintenant on souhaite tester différentes méthodes de machine learning. Nous allons commencer par la transformation des données quantitatives ('sex', classes', etc.) grâce à ***.get_dummies()***.

Finalement, nous allons générer un dataset d'entraînement et autre de test. Pour cela, on sépare les colonnes avec les variables d'entrée et la colonne à prédire. Ensuite, on générera les datasets en réservant 25% des données pour la partie test.

```
from sklearn.model_selection import train_test_split
titaX = intercept.drop(intercept.columns[-1],axis=1)
titaY = intercept[intercept.columns[-1]]
X_train, X_test, Y_train, Y_test = train_test_split(titaX, titaY,
test_size=0.25)
```

Partie 12 – Régression Linéaire simple

On démarrera par une régression linéaire simple. Grâce à Scikit-Learn, on a déjà une classe LinearRegression() tout prête à être utilisée. Vous pouvez faire une régression multivariate, en utilisant l'ensemble des colonnes.

On affichera les "poids" de chaque variable puis on fera des prédictions avec le dataset de test, avant d'afficher la comparaison entre les résultats attendus et les prédictions sous forme graphique.

Partie 13 – K plus proches voisins et Random Forests

Comme les prédictions avec la régression linéaire sont loin d'être précises, on essayera avec deux autres méthodes, les KNN et les RandomForests.

Pour finir, vous pouvez afficher différentes métriques (MSE, MAE, RMSE).