Exercices d'application

Numpy

Utilisation standard de numpy

Créer une fonction `dummy` qui :

- prend en argument un entier positif non nul N (`assert` pour le vérifier)
- retourne un tableau avec 1 axe et deux éléments. Le 1er contient la somme des N premiers nombres paires, l'autre les N premiers nombres impaires.

La solution ne doit contenir aucune boucle et aucun appel de fonctions non issues de numpy.

Faire deux implémentations :

- une générant un tableau de taille N
- l'autre de taille N*2

Calcul d'un histogramme

Écrire une fonction simplifiée de calcul d'histogrammes. Voici un exemple d'utilisation :

```
nb_samples = 10
data = np.random.randint(0,20, nb_samples)
histo = histogram(data, 0, 20, 5)
print(data)
print(histo)
print(np.sum([couple[1] for couple in histo]))
assert np.sum([couple[1] for couple in histo]) == nb_samples
```

Et sa sortie :

```
[ 5  0  9 11 12 15  7 14 12  3]
[(0.0, 2), (4.0, 2), (8.0, 2), (12.0, 4), (16.0, 0)]
10
```

Manipulation d'indices dans une application réaliste

Le `Leave One Out` est une technique utilisée pour faire de la validation croisée lorsque le jeu de données a peu d'éléments. Elle consiste à prendre N-1 données du jeu de données pour apprendre le modèle et utiliser la donnée restante pour le tester. L'opération est effectuée N fois pour que chaque exemple soit utilisé 1 fois pour le test.

Écrire la classe python qui me permette d'effectuer les traitements suivants (il faut implémenter une méthode `__iter__`):

```
nb_samples = len(dataset)
assert nb_samples == len(groundtruth)
loo = LeaveOneOut(nb_samples)
result = np.empty((nb_samples,))

for train_idx, test_idx in loo:
   X_train = dataset[train_idx]
   y_train = groundtruth[train_idx]
   X_test = dataset[test_idx]

   clf = Classifier()
   clf.train(X_train, y_train)
   pred = clf.predict(X_test)

   result[test_idx] = pred
```

Écrire les tests de validation de cette classe.

Pandas

Algorithmique basique de gestion de données

Le code suivant :

```
from sklearn.datasets import *
data = load_digits()
X, y = data["data"], data["target"]
assert len(X) == len(y)
```

permet de récupérer dans `X` des vecteurs qui représent des nombres manuscrit en niveau de gris et `y` le nombre correspondant.

- Créer un `DataFrame` qui contient autant de colonnes que la dimension des vecteurs de `X`. Les nommer les `pixel_<i>`.
- Ajouter une colonne `groundtruth` qui contient la vérité terrain.
- Ajouter une colonne `random` qui ne contient que des nombres aléatoires et n'est présente que pour complexifier (et rendre plus réalistes) les traitements suivants.
- Créer, programmatiquement, un dataframe par valeur de vérité terrain (regarder la fonction `groupby`). Calculer le vecteur moyen pour chacun d'entre eux et stocker dans un nouveau dataframe la valeur maximum,minimum,moyenne,écart type de chacun d'entre eux avec la vérité terrain. L'attribut `values` du dataframe peut être utile, tout comme la création de dataframe depuis un dictionnaire. S'assurer que l'ordre des colonnes est le suivant : min, max, mean, std, count, gt.

Analyse d'un jeu de données simple

Cet exercice est une adaptation du suivant : https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/problem12.html.

Récupérer le fichier suivant https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv qui contient la base de données des naufragés du Titanic.

- Utiliser `head` pour deviner le format de stockage des données.
- Charger ce fichier en utilisant les fonctionnalités de pandas
- Calculer la probabilité conditionnelle qu'une personne survive (`Survived`) en fonction de son genre (`Sex`) et de sa classe de passager (`Pclasse`). Les différentes valeurs doivent être obtenues de façon programmatique.

- Calculer la probabilité qu'un enfant de 10 ans ou moins en 3ième classe survive.
- Quel est le prix moyen d'un billet en fonction de la classe. Penser à `groupby`.

Chargement de données.

Lecture de fichier `CSV`

- Fichier : https://www.data.gouv.fr/fr/datasets/emissions-de-co2-et-de-polluants-des-vehicules-commercialises-en-france/ [2014] Emissions de polluants, CO2 et caractéristiques des véhicules commercialisés en France
- Exercice : écrire un parseur (sans utiliser de dépendances) qui génère une liste de dictionnaires représentant les données du fichiers. Parcourir la liste pour répondre aux question suivantes :
- Quel véhicule est le plus puissant ?
- Quel véhicule est le plus gourmand en ville ?
- Hints :
- Utiliser `file` pour tenter de deviner l'encodage du fichier.
- Utiliser `head` pour identifier les colonnes du fichier et leur type.
- Utiliser l'IDE pour faire une liste de chaine de caractères contenant le nom de ces colonnes.

Bien entendu, dans la vraie vie, on utilisera un meilleur parseur (celui fourni par `pandas` par exemple).

Lecture de fichier `JSON`

- Fichier : https://www.data.gouv.fr/fr/datasets/departements-et-leurs-regions/_ departements-region.json
- Exercice : écrire un parseur (en utilisant la dépendence `json`) qui stocke la base de données dans une `Base` qui contient les `Region` qui contient des `Departement`. Ajouter les méthodes nécessaires pour :

- Afficher la liste des régions par ordre du nombre de département

Lecture de fichiers `XML`

- Fichier : https://www.data.gouv.fr/fr/datasets/table-de-composition-nutritionnelle-des-aliments-ciqual/ Table Ciqual 2017 en Français au format XML -Exercice : écrire un parseur (en utilisant la dépendence `etree`) qui stocke la base de données dans un ensemble d'objets cohérents pour cet ensemble de fichiers. Générer ensuite un rapport au format `markdown` qui résumé cette base. Titre niveau 1 = Groupe, Titre niveau 2 = aliment. Ne modéliser que le groupe et pas les sous-groupes ou sous-sous groupes.

Récupération d'information dans une page HTML

- Exercice écrire un parseur qui récupère la valeur indiciaire du cours de bourse de votre choix sur sa page boursorama (en utilisant les dépendances `BeautifulSoup` et `requests`).

Intégration avec des DataFrames

- Refaire l'exercice de lecture de fichier XML (il est préférable de copier-coller le code de l'exercice précédent pour récupérer les morceaux communs). Au lieu de stocker les informations dans des objets ad-hocks, les stocker dans des dataframe différents.
- Interroger les DataFrame pour répondre aux question suivantes :
    - Quel est le code de l'aliment Vodka ?
    - Quel est le nom du groupe de l'aliment Vodka ?
    - Quel sont ses composants ?

Extraction de données structurées

Soit le graphe d'amitiés entre différents individus représenté de la façon suivante :

```
users = [
    {"id":0, "name": "A"},
    {"id":1, "name": "B"},
    {"id":2, "name": "C"},
    {"id":3, "name": "D"},
    {"id":4, "name": "E"},
    {"id":5, "name": "F"},
    {"id":6, "name": "G"},
    {"id":7, "name": "H"},
    {"id":8, "name": "I"},
    {"id":9, "name": "J"},
]

friendships = [
    (0,1), (0,2), (2,3), (1,3),(3,4),(4,5),(5,6),(5,7),(6,8),(7,8),
(8,9)
]
```

- Quel est le nombre moyen de connection entre individus ?
- Qui a le plus d'amis ?
- Écrire une fonction qui permet de récupérer les amis des amis de l'individu concerné sous la forme d'un dictionnaire (clé=identifiant d'un ami d'ami et valeur=nombre d'occurences)

De la donnée brute à la donnée exploitable

Présentation

Pour réaliser cet exercice, vous allez devoir à la fois manipuler des concepts illustrés lors des exercices des séances précédentes, ainsi d'autres non acquis. Vous allez devoir prendre le temps de lire la documentation des différentes bibliothèques.

Vous devez créer un modèle de prédiction d'usage d'un système de vélos en libre service. Pour celà, vous avez collecté pendant plusieurs mois les données d'utilisation du système de la ville de Parme ainsi que des données météo associées. Vous devez nettoyer ces données afin qu'elles soient directement exploitables par la bibliothèque de machine learning de votre choix.

Les données sont accessibles à cette ici.

Travail à réaliser

Vous êtes encouragés à travailler en binômes. Écrivez le code qui permet de :

- lire les données brutes directement depuis le fichier zip sans étape de décompression intermédiaire (voir les modules `zipfile` et `pandas`);
- supprimer les données abérentes (dates/heures impossibles, données invalide, erreur de collecte) ;
- normaliser le nom des stations (les noms peuvent avoir changé au cours du temps) grâce au fichier `brut/bicincitta_parma_summary.csv` ;
- ré-échantilloner les données pour avoir un enregistrement toutes les 10 minutes(au lieu de 5 minutes (voir la méthode `resample` des `DataFrame` s);
- fusionner les données vélo et météo (voir la méthode `merge` des `DataFrame` s);
- stocker les données transformées sous la forme suivante :
  - 1 dossier différent par station (soit 24 dossiers) (voir `groupby`);
  - Dans chaque dossier, 1 fichier par sous-série temporelle au format `.csv.gz` . Une sous série temporelle contient tous les exemples consécutif d'une station. Le nombre de sous séries dépend du nombre de "trous" dans les données (*i.e.*, le nombre de fois où il y a plus de 10 minutes d'écart entre deux enregistrements consécutifs). Il s'agit probablement de la partie la plus compliquée de ce travail. À vous de trouver comment détecter cette présence de trous.
- Il est préférable, mais pas obligatoire, d'utiliser une barre de progression pour donner une idée de la quantité de calcul restant et de l'attente.

Le contenu généré dans un fichier d'une sous-série doit ressembler à (vous en déduirez les informations à conserver depuis les données brutes):

```
Timestamp;Station;Bikes;Slots;Total;Status;Humidity;Pressure;Rain;Wir
2018-01-25 11:20:00;01. Duc;3;5;8;Mist;81.0;1028.0;{};240;1.5;{};5.
7
2018-01-25 11:30:00;01. Duc;3;5;8;Mist;81.0;1028.0;{};240;1.5;{};5.
69
2018-01-25 11:40:00;01. Duc;3;5;8;Mist;81.0;1028.0;{};240;1.5;{};5.
69
```

```
2018-01-25 11:50:00;01. Duc;3;5;8;Mist;81.0;1028.0;{};240;1.5;{};6.
64
2018-01-25 12:00:00;01. Duc;3;5;8;Mist;81.0;1028.0;{};240;1.5;{};5.
78
2018-01-25 12:10:00;01. Duc;3;5;8;Mist;81.0;1028.0;{};240;1.5;{};5.
78
2018-01-25 12:20:00;01. Duc;3;5;8;Mist;75.0;1027.0;{};290;2.1;{};6.
47
2018-01-25 12:30:00;01. Duc;3;5;8;Mist;75.0;1027.0;{};290;2.1;{};6.
46
2018-01-25 12:40:00;01. Duc;3;5;8;Mist;75.0;1027.0;{};290;2.1;{};6.
44
2018-01-25 12:50:00;01. Duc;3;5;8;Mist;75.0;1027.0;{};290;2.1;{};8.
0
2018-01-25 13:00:00;01. Duc;3;4;7;Mist;75.0;1027.0;{};290;2.1;{};6.
79
2018-01-25 13:10:00;01. Duc;3;4;7;Mist;75.0;1027.0;{};290;2.1;{};6.
79
```

Format des données source

Le format des données vélo est le suivant :

```
date;Station;Status;Nombre de vélos disponibles;Nombre
d'emplacements disponibles
```

Un status de 1, indique qu'il n'y a pas eu de soucis lors de la collecte. Voici un extrait de fichier :

```
2014-11-14 09:35:38;Duc;1;4;5
2014-11-14 09:35:38;Ospedale Maggiore;1;2;7
2014-11-14 09:35:38;Traversetolo;1;2;7
2014-11-14 09:35:38;Campus Chimica;1;4;5
2014-11-14 09:35:38;Stazione FF.SS.;1;9;10
2014-11-14 09:35:38;Ponte di Mezzo;1;4;6
```

Le format des données climat est le suivant :

```
Timestamp;Status;Clouds;Humidity;Pressure;Rain;"WindGust;WindVarEnd;W
```

Il n'y a pas besoin de comprendre le sens des colonnes pour effectuer le travail. Voici un extrait de fichier ; vous pouvez remarquer que le type de données est variable :

```
2014-11-14 09:35:38;clouds;40;100;1013;{u'3h': 0};None;None;None;
200.504;0.84;{};9.0;9.0;9.0
2014-11-14 09:45:05;mist;40;100;1014;{u'3h': 0};None;None;None;200.
504;0.84;{};10.0;10.0;10.0
```

- Pour des raisons d'efficacité, il est préférable de travailler avec des extraits de données lors de la phase de développement
- Vous pouvez effectuer le nettoyage en plusieurs étapes intérmédiaires avec sauvegarde sur disque
- Utilisez les bibliothèques de votre choix. La solution a été rédigée avec les imports suivants :

```
import os
import sys
import shutil
import pandas as pd
from dateutil import parser
import traceback
import zipfile

from joblib.memory import Memory
```

- La partie apprentissage ne nous intéresse pas du tout pour ce projet.
- Le résultat obtenu sera utilisé dans la suite des séances.

Exercices supplémentaires pour les curieux

- Numpy/matplotlib https://faculty.math.illinois.edu/~shahkar2/cbmg/numpy-exercises.html
- Dataframe https://www.machinelearningplus.com/python/101-pandas-exercises-python/
- Regardez ce que vous pouvez faire avec les jeux de données suivants :
    - Trajets de vélos NYC https://www.citibikenyc.com/system-data

- Trajets de taxi NYC https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page