Exercise 4: Python: libraries and FP

In this exercise you will create your first library and practice with more functions. For this exercise divide yourself in groups of 2-3 students (same or different from previous homework).

For this project each of the components of the group should do at least one commit together with a Pull Request. Pay attention to the commit messages and try to follow the best practices seen in class.

1. Create a new virtual environment for this project.
2. Create a repository in the account of one of the components of the group.
3. Clone the repository locally each of the components of the group.
4. Complete the exercises in hw4.py file and commit the solutions.
5. Create a Python library following the structure seen in class. (Needless to say, use different names to the folders and files that are representative of the purpose of the code they contain).
6. Finally, you are going to perform some data analysis on the data from "sample_diabetes_mellitus_data.csv". Imagine you are in a project that tries to predict whether some patients have diabetes mellitus. In order to do so, you create a library that will be used for your team in order to perform the analysis and make predictions. Since you want your colleagues to make the code as understandable as possible (and earn the highest grade possible), follow the best practices learnt during the lessons. Try to modularise the steps into functions (you can potentially merge g and h) and to separate the functions into files/folders in a way that makes the most sense to the team. For the purpose of this exercise, you are going to follow the next steps:
   a. Load the data.
   b. Split the data between train and test. (you can use train_test_split from sklearn or any other way)
   c. Remove those rows that contain NaN values in the columns: age, gender, ethnicity.
   d. Fill NaN with the mean value of the column in the columns: height, weight.
   e. Generate dummies for ethnicity column (One hot encoding).
   f. Create a binary variable for gender M/F.
   g. Train a model (for instance LogisticRegression or RandomForestClassifier from sklearn) in the train data. Use as features the columns: 'age', 'height', 'weight', 'aids', 'cirrhosis', 'hepatic_failure', 'immunosuppression', 'leukemia', 'lymphoma', 'solid_tumor_with_metastasis'. Use as target the column: 'diabetes_mellitus'
   h. Predict the targets for both the train and test sets and add the prediction as a new column (use predict_proba from the model to get the predicted probabilities) name the new column something like predictions.
   i. Compute the train and test roc_auc metric using roc_auc_score from sklearn.
7. Create a notebook that imports the functions and executes the previous steps in order to compute the metrics. Notice that the purpose of the exercise is to practice programming skills (specially regarding functions usage, DRY, code readability and best practices), the creation of a python library and the use of Git (getting used to it and practice collaboration among teams). The actual results on the metric, the appropriateness of the split, etc. are not going to be evaluated.
8. Share the link to the repository by turning in the assignment and let my user (Icedgarr) have access to it.

Some picks from the **Zen of Python**

Explicit is better than implicit.

Readability counts.

Simple is better than complex.

Flat is better than nested.

Special cases aren't special enough to break the rules.