Scientific Programming with Python – Exercise 1

Introduction

You are tasked with creating the DataSummary class. This class will be able to hold data records collected from a JSON file and return summary information as described in the following section. The JSON will contain records, each of which will have a subset of features from a possible list of features. You will also be given a CSV file that will contain meta-information regarding those features and their data types.

You will create a module named data_summary which will contain the DataSummary class. Ensure you use these names as they will be imported in this manner. Attached is an example Python code that will be run to test your module. Note that this is only an example which shows part of the tests that will be run. Your class must support all functionalities described in the API section. For this exercise you are not allowed to import any external modules except for "csv", "os" and "json".

DataSummary API

- Constructor: receives a JSON file as the first parameter (named "datafile") and a CSV file as the second parameter (named "metafile"). If one of these files does not exist or is not sent to the constructor, an Exception(ValueError) must be raised.
    - If a feature does not exist for a record in the JSON but is defined by the CSV file, set it to None (python NoneType object).
    - If a feature exists for a record in the JSON but is not defined by the CSV file, it should be ignored (not saved as part of the DataSummary object).
- DataSummary will support position-based indexing (read only) using the '[]' operator. DataSummaryObject[i] will return the "i^{th}" record (based on the original order in the JSON) as a dictionary object, with its features as the dictionary keys. Any feature from the feature list (as defined by the CSV file) that does not exist for the record will have a value of None (Python NoneType). (i.e. the "row" of that record if this were a table). An out of bounds index will raise an Exception(IndexError).
    - Note – make sure the dictionary returned is a copy of the data so that DataSummary itself cannot be modified.
- DataSummary will support key-based indexing (read only) using the '[]' operator. DataSummaryObject[feature] will return a list of all values (including duplicates) for that feature. (i.e. the "column" of that feature if this were a table). If the "feature" key does not match a feature of the data, an Exception(KeyError) will be raised.
    - Note – make sure the list returned is a copy of the data so that DataSummary itself cannot be modified.
- DataSummary objects will support the following functions, which will accept a "feature" parameter as input. If the "feature" key does not match a feature of the data, an Exception(ValueError) will be raised.
  Note that only the count, unique, mode and empty functions will be supported for categorical features. Attempting to call one of the other functions on a categorical feature should raise an Exception(TypeError).

    - .sum(feature): return the sum of all values in the feature.
    - .count(feature): return the number of non-None values in the feature.
    - .mean(feature): return the average over all values in the feature (None values are not counted)
    - .min(feature): return the minimal value in the feature.
    - .max(feature): return the maximal value in the feature.
    - .unique(feature): return a list of unique values in the feature, in ascending order (alphabetical for categorical features).
    - .mode(feature): return a list of the values which appear the most in the feature (None values are not counted).
    - .empty(feature): return the number of None values in the feature.
- DataSummary will allow creating a new CSV file based on the input JSON file and CSV meta-information. DataSummaryObject.to_csv(filename,delimiter=',') will create a csv file with the given filename, having the features as column names and each row as a record from the original JSON file. By default, the delimiter will be a comma, however the "space" character and any of the characters in the following string should also be supported: ". : | - ; # *". If an unsupported delimiter is requested, the default comma will be used.

File Formats

CSV:

The first row is a comma separated list of features.

The second row is a comma separated list of data types. For each feature this can be int, float or string.

JSON:

The top hierarchy is a single key "data" whose value is a list of records. Each record is a JSON object containing key-value pairs matching the features defined by the CSV file. Note that not all records contain all features, and some may contain features that do not exist in the CSV feature list.

Validation

Assume valid input except for validations stated inside each function definition in the API.

Submission

You will submit a single data_summary.py file with your names and IDs commented at the top.

You will add documentation (in python format) to each function and the class giving a short description and indicating the input parameters and returned values. You may write helper functions inside the module, make sure to document them and that they are not part of the class functions.

Write clean and efficient code. Points will be deducted for poorly written code or implementations with unreasonable runtimes.