Learning goals: Numpy, Pandas, Databases, structured text file I/O

EXERCISES

(1) Create a module, CSVProcessor. It should contain functions for loading CSV data from an external file (titanic.csv), calculating total number of columns, calculating total number of rows and filling missing values in any column with zero. Use Pandas read_csv and other Pandas and Numpy functions. Import this module into another program and demonstrate invoking these methods.

(2) 2. Create a module, JSONProcessor. It should contain functions for loading JSON data from an external file and printing JSON data. The JSON file should contain following player details. Create a JSON file with this information.

```
{
   {
    \player_name": \Shubham"
    \player_email": \shubham@abc.org"
    \player_score: 45
    \man_of_the_match": false
   },
   {
    \player_name": \Rohit"
    \player_email": \rohit@abc.org"
    \player_score: 75
    \man_of_the_match": false
   },
   {
    \player_name": \Virat"
    \player_email": \virat@abc.org"
    \player_score: 100
    \man_of_the_match": false
   }
]
}
```

Load this file into a dictionary using the JSON module. Set the `man_of_the_match` field to `True` for a player who has scored the maximum score among all players. Write back this information into a new JSON file.

(3) Create a base class, `CDataProcessor` with two properties - samples and features - and a method `PrintDatasetInfo()`. The two properties are initialized using the number of rows and columns of a Pandas dataframe that is passed to the constructor during object creation. The `PrintDatasetInfo()` method should print the number of samples and features.

Derive a new class, `CCSVProcessor` from the `CDataProcessor` class. This derived class should have two properties - `filename` and `dfData` - The filename is initialized to path of the CSV file specified during object creation. The `dfData` is initialized to empty dataframe. The `CCSVProcessor` class should contain two methods - `LoadData()` and `ConvertToJSON()` - `LoadData` should load the CSV data into a `dfData` property of this class (use Pandas read_csv method). It should also invoke its parent class `__init__`

method passing the dfData, so that, parent's samples and features are populated correctly. `ConvertToJSON()` should create a new JSON file using the `dfData` (use Pandas to_json method)

Derive a new class, `CJSONProcessor` from the `CDataProcessor` class. This derived class should have two properties - `filename` and `dfData` - The filename is initialized to path of the JSON file specified during object createion. The dfData is initialized to empty dataframe. The `CJSONProcessor` class should contain a method - `LoadData()` - `LoadData` should load the JSON data into a dfData property of this class (use Pandas read_json method). It should also invoke its parent class `__init__` method passing the dfData, so that, parent's samples and features are populated correctly.

a. Create an instance object of CCSVProcessor using the file titanic.csv. Load the data and invoke `PrintDatasetInfo()` to print the number of samples and features in this dataset.
b. Create an object of CCSVProcessor using the file `ODI-Batting_Cricket_Analytics.csv`. This file is copied in my Google drive code folder. Load the data and invoke `ConvertToJSON()` method to create a new `ODI-Batting_Cricket_Analytics.JSON file`.
c. Create an object of `CJSONProcessor()`, load the data and invoke `PrintDatasetInfo()` to print the number of features and columns of this dataset.

(4) Database exercise:

(i) Create a database called `school.db`. You are going to build a simple student marks management system using `SQLite3`. Inside this databse, create a table called `students` with the following columns: `id` of type INTEGER, `name` of type TEXT, `subject` of type TEXT and `marks` of type INTEGER.

(ii) Insert the following records into the `students` table:

id	name	subject	marks
1	Ravi	Math	85
2	Meena	Science	90
3	Arjun	English	78
4	Priya	Math	92
5	Suresh	Science	65

(iii) Write SQL queries using Python to a) display all records in the table, b) display the names of students who scored above 80 marks, c) display the average marks in each subject.

(iv) Update "Suresh's" science marks from 65 to 75.

(v) Delete the record of the student who scored the lowest marks in English.

(vi) Add a new column called 'grade' to the table, and assign "A" for marks >= 85, "B" for marks between 70-84, and "C" for marks< 70.