Assignment

771768 - Introduction to Programming for Artificial Intelligence and Data Science

Customer Data Pre-processing

(This assignment is worth 100% of the total marks for this module)

This assignment should be submitted on **CANVAS**

Context

This assignment is designed to evaluate your ability to read and write file formats of common types used in Data Science, and to manipulate complex data into different representations. The tasks provided here are indicative of Data Pre-processing workloads which are common to all Data Science projects. The techniques learned and evaluated in this module will prepare you for further theoretical and applied topics later on in the programme, where you will further develop your skills.

This assignment makes use of an extensive collection of mocked data. These have been generated with some resemblance to real world values and distributions, including some relations between data elements.

Whilst teaching, both asynchronous and synchronous, stops for this module by Teaching Week 4, ad-hoc support will be available until submission of the assignment on the MS Teams site.

Background

You have been given a collection of data from a company wishing to process its customer records for business purposes (**acw_user_data.csv**). The existing systems in-place at the company only export to a CSV file, and this is not in an appropriate format for analysis. You have been given the task of preparing this data for further analyses by your colleagues within the company, including representation changes, filtering, and deriving some new attributes / metrics for them.

These data include attributes such as first name, second name, credit card number, marital status, and even contains data on the customer's car.
The number of records provided is significant, and therefore it is expected that solutions are robust to varying types of data, and varying values, offering a programmatic solution.

Tasks

Data Processing (70%)

Using standard python (No pandas / seaborn) with default libraries (os, sys, time, json, csv, …) you have been given the following tasks:

1. Read in the provided ACW Data using the CSV library.

2. As a CSV file is an entirely flat file structure, we need to convert our data back into its rich structure. Convert all flat structures into nested structures. These are notably:
   a. Vehicle - consists of make, model, year, and type
   b. Credit Card - consists of start date, end date, number, security code, and IBAN.
   c. Address - consists of the main address, city, and postcode.

   For this task, it may be worthwhile inspecting the CSV headers to see which data columns may correspond to these above.

   Note: Ensure that the values read in are appropriately cast to their respective types.

3. The client informs you that they have had difficulty with errors in the dependants column. Some entries are empty (i.e. " " or ""), which may hinder your conversion from Task 2. These should be changed into something meaningful when encountered.

   Print a list where all such error corrections take place.

   E.g. Problematic rows for dependants: [16, 58, 80, 98]

4. Write all records to a **processed.json** file in the JSON data format shown in the appendix to this document. This should be a list of dictionaries, where each index of the list is a dictionary representing a singular person.

5. You should create two additional file outputs, **retired.json** and **employed.json**, these should contain all retired customers (as indicated by the retired field in the CSV), and all employed customers respectively (as indicated by the employer field in the CSV) and be in the JSON data format.

6. The client states that there may be some issues with credit card entries. Any customers that have more than 10 years between their start and end date need writing to a separate file, called **remove_ccard.json**, in the JSON data format. The client will manually deal with these later based on your output. They request that you write a function to help perform this, which accepts a single row from the CSV data, and outputs whether the row should be flagged. This can then be used when determining whether to write the current person to the **remove_ccard** file. Note the dates are shown in the format used on credit cards which is "MM/YY".

7. You have been tasked with calculating some additional metrics which will be used for ranking customers. You should create a new data attribute for our customers called “Salary-Commute”. Reading in from **processed.json**:
    a. Add, and calculate appropriately, this new attribute. It should represent the Salary that a customer earns, per Km of their commute.
        i. Note: If a person travels 1 or fewer commute Km, then their salary-commute would be just their salary.
    b. Sort these records by that new metric, in ascending order.
    c. Store the output file out as a JSON format, for a **commute.json** file.

Data Visualisation (20%)

Using Pandas and Seaborn

Your client wishes to understand the data they have on their customers a bit more by use of visualisations. With use of Pandas and Seaborn read in the original CSV file provided with the assignment.

1. Obtain the Data Series for Salary, and Age, and calculate the following:
    a. Mean Salary
    b. Median Age
2. Perform univariate plots of the following data attributes:
    a. Age, calculating how many bins would be required for a bin_width of 5.
    b. Dependents, fixing data errors with seaborn itself.
    c. Age (of default bins), conditioned on Marital Status
3. Perform multivariate plots with the following data attributes:
    a. Commuted distance against salary.
    b. Age against Salary
    c. Age against Salary conditioned by Dependants
4. Your client would like the ability to save the plots which you have produced. Provide a Notebook cell which can do this. You should execute this cell and attach the generated plots with your submission.

Code Presentation (10%)

The code you produce to solve the above tasks should make good use of structure, logic, and commenting to be clear and robust. Variable names should be thoughtfully considered, and appropriate for purpose. Ensure that scope conflicts are avoided, and that variable names don’t leak into other areas of code. You should consider how the structures covered throughout the module may be used. Similarly, you should be mindful of error handling where appropriate.

Use of Markdown Cells is advised to keep clear distinction between Tasks. When writing your algorithms for solving the above, it may be useful to keep in mind reusability of code, reducing the amount of boilerplate code, and duplicated code.

Grading and Deliverables

Submission is to be a complete ipynb notebook, alongside any file outputs your program generates (E.g. **processed.json**, **retired.json**, **employed.json**, **remove_ccard.json**, **commute.json**, including any seaborn figures saved).

The Notebook should contain all output cells (i.e. Showing the program output from when you executed the code to generate the attached files mentioned above).

These should be **submitted as a single ZIP file** to the appropriate upload zone on Canvas.

The following grading criteria will be applied to your submissions. Each section is presented as a mark out of 100, representing quality boundaries of Pass, 2:2, 2:1, First, and Upper First Class. These are then weighted to produce an overall score for the assignment out of 100.

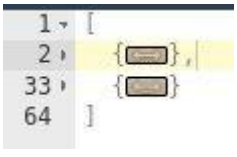
Processed JSON Example


```
[
  {
    "first_name": "Amanda",
    "second_name": "Hussain",
    "age": 68,
    "sex": "Female",
    "retired": false,
    "marital_status": "single",
    "dependants": 1,
    "salary": 49142,
    "pension": 0,
    "company": "Curtis, Short and Bell",
    "commute_distance": 12.86,
    "Vehicle": {
      "make": "Toyota",
      "model": "Outback",
      "year": "2016",
      "category": "Van/Minivan"
    },
    "Credit Card": {
      "start_date": "09/15",
      "end_date": "12/31",
      "number": 4541066368266,
      "ccv": 125,
      "iban": "GB76ODNO24203982154576"
    },
    "Address": {
      "street": "07 Ben oval",
      "city": "Deborahburgh",
      "postcode": "IM8R 8NQ"
    }
  },
  {
    "first_name": "Sean",
    "second_name": "Cook",
    "age": 59,
    "sex": "Male",
    "retired": false,
    "marital_status": "single",
    "dependants": 5,
    "salary": 61334,
    "pension": 0,
    "company": "Bray-Owen",
    "commute_distance": 14.95,
    "Vehicle": {
      "make": "Chrysler",
      "model": "PT Cruiser",
      "year": "1993",
      "category": "Sedan"
    },
    "Credit Card": {
      "start_date": "05/12",
      "end_date": "12/23",
      "number": 4851307868757239,
      "ccv": 9765,
      "iban": "GB95CVDH40711859915263"
    },
    "Address": {
      "street": "838 Thomas passage",
      "city": "Millston",
      "postcode": "M1 7QS"
    }
  }
]
```


The above is an example of JSON formatted output from processed. This is a List [ ], where each index is a dictionary representing a person. Each person is a dictionary, whereby we have multiple key: value pairs, for multiple data types. These include sub-dictionaries “Vehicle”, “Credit Card”, “Address”.

Any JSON output should be structured this way. Where each ‘row’ from your CSV represents a person in this instance.

The above example was generated with test data which is NOT in the ACW data provided.



UNIVERSITY OF HULL

771768 Introduction to Programming AI & Data Science - CRG

ACW - Customer Data Preprocessing

Criterion	Up to 40%	Up to 50%	Up to 60%	Up to 70%	Up to 80%	Up to 100%
Data Processing Converting from CSV to JSON output, with multiple manipulation and filtering steps. (70%)	The delivered package could not be deployed to process data without significant changes.	Basic processing of the CSV data to JSON output is provided, but may contain errors when deployed	In addition to previous. The JSON representation (Task 2) is attempted, but may not be fully correct. Additionally, tasks may be attempted or do not run correctly when deployed.	In addition to previous. JSON Representation is correct with no errors present. All Tasks are attempted, but some may contain errors in either execution, or in the contents of file output. Output to file must be valid, but may contain missing data/records.	In addition to previous. All Tasks are completed without error, with all output JSON files provided. Casting for all values processed are done correctly, with error-handling appropriately. No unused variables are present.	In addition to previous Techniques to manipulate data are well-considered and algorithms designed to be as robust to potential data changes. Solution relies minimally on code duplication, and additional I/O opening/closing.

Demonstrate the use of Python Jupyter Notebooks towards the task of preprocessing data for Data Science processes. (10%)	The Python code is unable to be deployed without significant changes.	A basic notebook is presented, which executes, but may not fully utilise multiple cells or any markdown features. Code may contain some errors which prevent successful operation.	In addition to previous. Code may function; however, scoping may not be carefully managed, with some variables unintentionally being used where they should not. Algorithms developed may contain spurious variables, or unused variables.	In addition to previous. Commenting is appropriate, and may make use of Markdown. Algorithms are generally well-executed, but may be inefficient in places. Scoping is correct, with variables named appropriately.	In addition to previous. Markdown Cells are used well to introduce tasks and explain developed code, in addition to code commenting. Additional commenting mechanisms may be attempted such as function documentation with multi-line strings. Developed algorithms to solve the tasks are efficient with careful consideration of the task and data structures used.	In add Code presented is efficient, and designed with consideration for future expansion of Data Preprocessing for the exemplar data. Additional commenting mechanisms are well-used for function documentation with multi-line strings. Additional language features have been utilised to improve code clarity, readability, etc.

Data Visualisation with Pandas and Seaborn. (20 %)	The data visualisation was only partially provided and the code could not be deployed to generate any further plots without significant changes.	Basic use of Pandas and Seaborn may be presented, but may be erroneous or incomplete in places. Statistical measures of Data Series or DataFrame are presented.	In addition to previous. Univariate plots are error free. Multivariate plots may be attempted, but contain errors.	In addition to previous. Multivariate plots are mostly complete Pandas manipulation is appropriate, with clear distinction between DataFrame and Series objects.	In addition to previous. All plots are performed error-free. Notebook contains code to save each figure presented.	In addition to previous. Excellent use of seaborn functionality, with good considerations for variable and memory usage. Additional parameters may be used to modify the plots to improve readability and information conveyance.

	All criteria are weighted as shown by the percentages indicated in the relevant criterion box.	