randstad

ARTIFICIAL INTELLIGENCE CHALLANGE

1. CHALLANGE

Randstad Italia, a world leader in the human resources sector, wants to develop an intelligent system able to classify its own ***job offer*** history (job offers).

Participants are required, given a **dataset** of job offers of about **2.000** records made available, to create a model that is able to classify the ***job descriptions*** of such offers automatically on the basis of specific labels (labels).

This dataset is therefore composed of 2 fields:
- ***Job description***: contains records describing the job offer;
- ***Label***: contains the label associated with the ad.

The reference labels are 5 and are listed below:
- ***Java Developer***;
- ***Web Developer***;
- ***Programmer***;
- ***System Analyst***;
- ***Software Engineer***.

N.B. The labels assigned to job offers are exclusive, which means that each offer is assigned exclusively to a specific category.

The database is provided in ".*csv*" format, already split into train and test set according to an 80/20 ratio and maintaining a balance between the labels.

2. FOLDER CONTENT

The content of the folder is as follow:

- ***main.py***: file with the source code;
- ***model.pickle***: file with the saved model;
- ***output.csv***: files with *predictions* on the test set;
- ***train_set.csv***: dataset for the training phase;
- ***test_set.csv***: model for the testing phase;
- ***README.pdf***: additional documentation to the work carried out.

3. ENVIRONMENT SETUP

Python (version *3.8.10*) has been chosen as the programming language for the execution of the challange. The Machine Learning library chosen for the training and testing phases is ***Scikit-Learn***. For the installation of this library please refer to this guide. All other libraries used are included by default in the ***Python Standard Library***.

4. CODE ANALYSIS

The code is structured as follows:

▪ *Import* of libraries used;
▪ *Functions* useful for code execution;
▪ *Executable code*.

With regard to imports, these are carried out as a preliminary operation. Specifically, the libraries used are ***csv*** (management of datasets and creation of files required for the solution of the challange), ***pickle*** (saving of generated models), ***os*** (management of the contents of the folder - avoid overwriting files that already exist) and ***Scikit-Learn*** (performing Natural Language Processing operations).

Listed below are the functions in the source code and their role:

▪ ***dataset_reader***: reading of the provided data. The function is designed to operate differently on training and testing data. This difference consists in making on the first of the two datasets an additional string manipulation operation indicated below:

```
[...]
record_job_offer = record_job_offer.replace('. ', '').replace('.', '')
[...]
```

With this operation the program "cleans" in a targeted way the data from the training file. This operation positively influences the results of the testing phase, increasing (albeit slightly) the values of precision, recall and F1-score. However, this effect shall be neutralised if the data on which the testing phase is carried out are also subjected to the same operation;

▪ ***save_vectorizer***: Generates a file "*vectorizer.pickle*" in which the model for the vectorization of the data extracted from the datasets is saved.
NOTE: Since this function is not explicitly required in the challange specification, it is disabled by default;

▪ ***save_model***: Generates a file "*model.pickle*" in which the model used for the training and testing phases is saved;

▪ ***output_file_generator***: Starting from the dataset on which the testing phase is performed, generate a file "*output.csv*" containing the fields *Job_description*, *Label_true* and *Label_pred*.

These fields contain the job advertisements on which predictions are made, the labels associated with them and the labels predicted by the program;

▪ ***metrics_output***: Print in the terminal the values of precision, recall and F1-score.
NOTE: Printing may result "messy" due to formatting of the text.

The code executed subsequently uses these functions as follows:

The datasets are read and the content is saved within the program, separating *Job description* from the *Label* associated with each of them.
Both training and testing data are "*vectorized*", that is translated from text strings into vectors that can be interpreted by the program.
NOTE: At this point in the program there are some commented code entries that correspond to the vectorizer save function. As mentioned above this function is disabled by default, but to enable it is sufficient to uncomment that portion of code.
At this point, through the specific model the program is "trained" on the data extracted from the training dataset (IMPORTANT: the model is imported as it is present in the project folder by default. Otherwise the training phase is performed within the program itself and the model is generated later).
Based on the operations carried out so far, the program calculates the values of precision, recall and F1-score related to testing data and reports them to the terminal.

5. MODEL USAGE

Inside the folder there is the file "*model.pickle*" for the reproduction of the results outside the program. This model may be imported as follows:

```
model = pickle.load(open('model.pickle', 'rb'))
```

To save and open the file you have chosen the *pickle* library. This library, in addition to being easy to use, offers the advantage of being pre-installed in Python, so it does not need preliminary operations in addition to the initial import.
In order to obtain results consistent with the program (in particular with precision, recall and F1-score values), however, it is necessary to vector the data in the same way as they were vectored in the source code. In fact, the optimal processing of information has a huge influence on the results obtained. To be able to use the vector model used inside the code it is possible to uncomment the portion of code and run the program: a file "*vectorizer.pickle*" will be generated inside the folder. This file can be uploaded to any other file in the same way as the file related to the model mentioned above:

```
vectorizer = pickle.load(open('vectorizer.pickle', 'rb'))
```

6. RESULTS

The results obtained through the work carried out in terms of **precision**, **recall** and **F1-score** are shown below.

PRECISION	
Java developer	0.8172043010752689
Web developer	0.925531914893617
Programmer	0.7281553398058253
System Analyst	0.9852941176470589
Software Engineer	0.8024691358024691

RECALL	
Java developer	0.8351648351648352
Web developer	0.8446601941747572
Programmer	0.78125
System Analyst	0.9054054054054054
Software Engineer	0.8666666666666667

F1-SCORE	
Java developer	0.8260869565217392
Web developer	0.883248730964467
Programmer	0.7537688442211056
System Analyst	0.943661971830986
Software Engineer	0.8333333333333334