API - Current

For the API I have currently solved the issue related to relative and absolute paths

Additionally, I do not think there is the need for local and remote CONF files. I managed to remove the sys locator by launching the main.py file (the app basically) through the following command:

```
python -m cic.main
```

Setting the root directory to `cic` .

To remove the sys addition for root project I also made the following changes:

```
SOLVED
I have used "python -m cic.main" to run the project, instead of
"python main.py". With -m cic.main the entire application is see
as a project, and each folder as a module, thus I have used abso
for module (e.g. instead of "from blueprints.cic_api import api_
"from cic.blueprints.cic_api import api_bp", thus with the cic.

cic is here considered the root directory of the project, so the
started within the "classifier" folder (one level above cic).
```

For me - README.md

To efficiently run the application it's necessary to enter within the correct folder through the `cd` command. The correct folder is the one placed one level above the root of the project, thus:

```
cd /Users/lorenzo/Desktop/CEC_API-TestCONF/classifier/
```

Indeed the structure is the following:

```
├── CHANGELOG.md
├── LICENCE
├── README.md
├── classifier
│   ├── ExampleAPI_Command.txt
│   ├── README.md
│   ├── cic
│   │   ├── __init__.py
│   │   ├── blueprints
│   │   │   ├── __init__.py
│   │   │   ├── cic_api.py
│   │   │   └── web_interface.py
│   │   ├── main.py
│   │   ├── sh...
│   │   ├── src
│   │   │   ├── __init__.py
│   │   │   ├── binary_classifiers.py
│   │   │   ├── data_processor.py
│   │   │   ├── metaclassifiers.py
│   │   │   ├── models
│   │   │   │   ├── NoSections...
│   │   │   │   ├── README.md
│   │   │   │   └── Sections...
│   │   │   └── predictor.py
│   │   ├── static...
│   │   └── templates...
│   └── requirements.txt
└── extractor
    ├── README.md
    └── cex
    └──...
```

Then, the other main change beside the use of absolute paths is in the API blueprint, in which I removed the specific path and used:

```
SRC_PATH = os.environ.get('SRC_PATH')

if not SRC_PATH:
    print("Error: The environment variable SRC_PATH is not set.'
    sys.exit(1)
```

With `os.environ.get('SRC_PATH')` the project takes directly the path from the environment. This should be good for both local and remote usage if correctly set. Indeed, when launching the application you need to specify the SRC_PATH in the following way:

```
SRC_PATH=/Users/lorenzo/Desktop/CEC_API-TestCONF/classifier/cic/
python main.py
```

The path above is my local path to the src directory, but the user has to specify his/her path. And for server it should be sufficient to do the same (I hope... **Ivan???**)

This, in turn led also to the following structural changes in predictor instantiation (model paths):

```
predictor_instance = Predictor(
            selected_mode,
            "allenai/scibert_scivocab_cased",
            "xlnet/xlnet-base-cased",
            [
                [
                    os.path.join(SRC_PATH, "models", "Sections",
                    os.path.join(SRC_PATH, "models", "Sections",
                    os.path.join(SRC_PATH, "models", "Sections",
                ],
                [
                    os.path.join(SRC_PATH, "models", "Sections",
                    os.path.join(SRC_PATH, "models", "Sections",
                    os.path.join(SRC_PATH, "models", "Sections",
                ],
            ],
            [
                [
                    os.path.join(SRC_PATH, "models", "NoSections
                    os.path.join(SRC_PATH, "models", "NoSections
                    os.path.join(SRC_PATH, "models", "NoSections
                ],
                [
                    os.path.join(SRC_PATH, "models", "NoSections
                    os.path.join(SRC_PATH, "models", "NoSections
                    os.path.join(SRC_PATH, "models", "NoSections
                ],
            ],
            os.path.join(SRC_PATH, "models", "Sections", "MetaCl
            os.path.join(SRC_PATH, "models", "NoSections", "Meta
        )
```

Shell commands and specs

Thus, the correct procedure to make the API work through terminal is:

1. Enter the correct folder (`cd` to the directory containing the project root `cic`, which is the one in which the `main.py` file is stored).
2. Run the `main.py` by specifying in the line also the `SRC_PATH` (path leading to the src folder).
3. Open another terminal window and run commands. The main command is composed as follows:

```
curl -X POST \
     -F 'file=@absolute/path/to/json_to_cls.zip' \
     -F 'mode=mixed' \
     "http://127.0.0.1:5000/api/classify" \
     --output results_file_prova_CEC.zip
```

   Breaking it down we have:

- `-F 'file=@absolute/path/to/json_to_cls.zip'` → path to the file (zip folder) to be classified
- `-F 'mode=mixed'` → specify the classification mode (WS, WoS, Mix)
- `"http://127.0.0.1:5000/api/classify"` → URL to which the `POST` request is sent
- `--output results_file_prova_CEC.zip` →
    - `-output`: Tells `curl` to write the output to a file instead of standard output.
    - `results_file_prova_CEC.zip`: The filename where the response will be saved. In this case it is saved by default in the working directory but, to change location just add relative/absolute path to the filename.

  Purpose of the command: saves the response from the server (which is expected to be a ZIP file) to `results_file_prova_CEC.zip` in the current working directory. (To see the current working dir use `pwd` within the terminal in which the command is written).

Output specs

The output will follow the same structure of the input folder, but will add also a `mainfest.json` file to the result. The manifest is organized as follows:

```
{
  "Initialization": {
    "Status": "Success",
    "Summary": {
      "Data Upload": "Success",
      "Classification Mode Selection": "Success",
      "Predictor Instantiation": "Success"
    }
  },
  "Data Loading": {
    "Files processing": {
      "Status": "Partial Procesing",
      "Description": "Not all the Files of the archive have 
      "Correctly loaded files": [
        "test_compressed_json1_1.json",
        "test_compressed_json3_1.json",
        "test_json_with_metadataZIPTEST.json",
        "test_compressed_json2_1.json",
        "test_compressed_json2_2.json"
      ],
      "Files generating errors": {
        ".DS_Store": {
          "Filename": ".DS_Store",
          "Error details": "File .DS_Store cannot be read."
        },
        "PresentationLetterSahar.docx": {
          "Filename": "PresentationLetterSahar.docx",
          "Error details": "File PresentationLetterSahar.doc
        }
      }
    }
  },
  "Single entries processing": {
    "Citation IDs Processing": {
      "Status": "Partial Procesing",
      "Description": "Not all the ids of the file have been 
      "Files entirely processed": [
        "test_compressed_json1_1.json",
        "test_compressed_json3_1.json",
        "test_compressed_json2_1.json",
        "test_compressed_json2_2.json"
      ],
      "Correctly loaded IDs": {
        "test_compressed_json1_1.json": [
          "1",
          "2",
          "3",
          "4",
          "5"
        ],
    "test_compressed_json3_1.json": [
      "1",
      "2",
      "3",
      "4",
      "5"
    ],
    "test_json_with_metadataZIPTEST.json": [
      "1",
      "3",
      "4",
      "5"
    ],
    "test_compressed_json2_1.json": [
      "1",
      "2",
      "3",
      "4",
      "5"
    ],
    "test_compressed_json2_2.json": [
      "1",
      "2",
      "3",
      "4",
      "5"
    ]
  },
  "Files generating errors": {
    "test_json_with_metadataZIPTEST.json": {
      "2": {
        "Citation ID": "2",
        "Error details": "Invalid JSON entry, the CITATI
      }
    }
  }
```

```
      }
    },
    "Classification": {
      "test_compressed_json1_1.json": {
        "Filename": "test_compressed_json1_1.json",
        "Status": "Success",
        "Summary": {
          "Classification process": "Success"
        }
      },
      "test_compressed_json3_1.json": {
        "Filename": "test_compressed_json3_1.json",
        "Status": "Success",
        "Summary": {
          "Classification process": "Success"
        }
      },
      "test_json_with_metadataZIPTEST.json": {
        "Filename": "test_json_with_metadataZIPTEST.json",
        "Status": "Success",
        "Summary": {
          "Classification process": "Success"
        }
      },
      "test_compressed_json2_1.json": {
        "Filename": "test_compressed_json2_1.json",
        "Status": "Success",
        "Summary": {
          "Classification process": "Success"
        }
      },
      "test_compressed_json2_2.json": {
        "Filename": "test_compressed_json2_2.json",
        "Status": "Success",
        "Summary": {
          "Classification process": "Success"
      }
    }
  }
}
```

As you can see it is divided into *operations*. Indeed we have:

1. **Initialization**

```
"Initialization": {
    "Status": "Success",
    "Summary": {
      "Data Upload": "Success",
      "Classification Mode Selection": "Success",
      "Predictor Instantiation": "Success"
    }
  },
```

2. **Data Loading**

```
"Data Loading": {
    "Files processing": {
      "Status": "Partial Procesing",
      "Description": "Not all the Files of the archive hav
      "Correctly loaded files": [
        "test_compressed_json1_1.json",
        "test_compressed_json3_1.json",
        "test_json_with_metadataZIPTEST.json",
        "test_compressed_json2_1.json",
        "test_compressed_json2_2.json"
      ],
      "Files generating errors": {
        ".DS_Store": {
          "Filename": ".DS_Store",
          "Error details": "File .DS_Store cannot be read
        },
        "PresentationLetterSahar.docx": {
          "Filename": "PresentationLetterSahar.docx",
          "Error details": "File PresentationLetterSahar.d
        }
      }
    }
  },
```

3. **Single entries processing**

```
"Single entries processing": {
    "Citation IDs Processing": {
      "Status": "Partial Procesing",
      "Description": "Not all the ids of the file have bee
      "Files entirely processed": [
        "test_compressed_json1_1.json",
        "test_compressed_json3_1.json",
        "test_compressed_json2_1.json",
        "test_compressed_json2_2.json"
      ],
      "Correctly loaded IDs": {
        "test_compressed_json1_1.json": [
          "1",
          "2",
          "3",
          "4",
          "5"
        ],
        "test_compressed_json3_1.json": [
          "1",
          "2",
          "3",
          "4",
          "5"
        ],
        "test_json_with_metadataZIPTEST.json": [
          "1",
          "3",
          "4",
          "5"
        ],
        "test_compressed_json2_1.json": [
          "1",
          "2",
          "3",
          "4",
          "5"
        ],
        "test_compressed_json2_2.json": [
          "1",
          "2",
          "3",
          "4",
          "5"
        ]
      },
      "Files generating errors": {
        "test_json_with_metadataZIPTEST.json": {
          "2": {
            "Citation ID": "2",
            "Error details": "Invalid JSON entry, the CIT
          }
        }
      }
    }
  },
```

4. **Classification**

```
"Classification": {
    "test_compressed_json1_1.json": {
      "Filename": "test_compressed_json1_1.json",
      "Status": "Success",
      "Summary": {
        "Classification process": "Success"
      }
    },
    "test_compressed_json3_1.json": {
      "Filename": "test_compressed_json3_1.json",
      "Status": "Success",
      "Summary": {
        "Classification process": "Success"
      }
    },
    "test_json_with_metadataZIPTEST.json": {
      "Filename": "test_json_with_metadataZIPTEST.json",
      "Status": "Success",
      "Summary": {
        "Classification process": "Success"
      }
    },
    "test_compressed_json2_1.json": {
      "Filename": "test_compressed_json2_1.json",
      "Status": "Success",
      "Summary": {
        "Classification process": "Success"
      }
    },
    "test_compressed_json2_2.json": {
      "Filename": "test_compressed_json2_2.json",
      "Status": "Success",
      "Summary": {
        "Classification process": "Success"
      }
```

```
        }
    }
```