**Introduction:**

I am working on trying to scrape, organise, clean and combine data for the Indian Administrative Services (IAS) Officers who were appointed from the year 1951 to 2019. This involves data extraction from these government websites:

* [DOPT](https://easy.nic.in/civilListIAS/YrCurr/FinalCL.asp)
* [Supremo](https://supremo.nic.in/KnowYourOfficerIAS.aspx)

These 2 websites primarily display the information about IAS officers like: Name, Date of Birth, Cadre, Identity No., etc. Even though these two websites provide different amount of data and in different time periods (DOPT: 1983-2019 & Supremo: 1951-2019), the overlapping information is consistent.

**Analysis of the websites:**

1. DOPT (*Hosted at* *easy.nic.in)*:

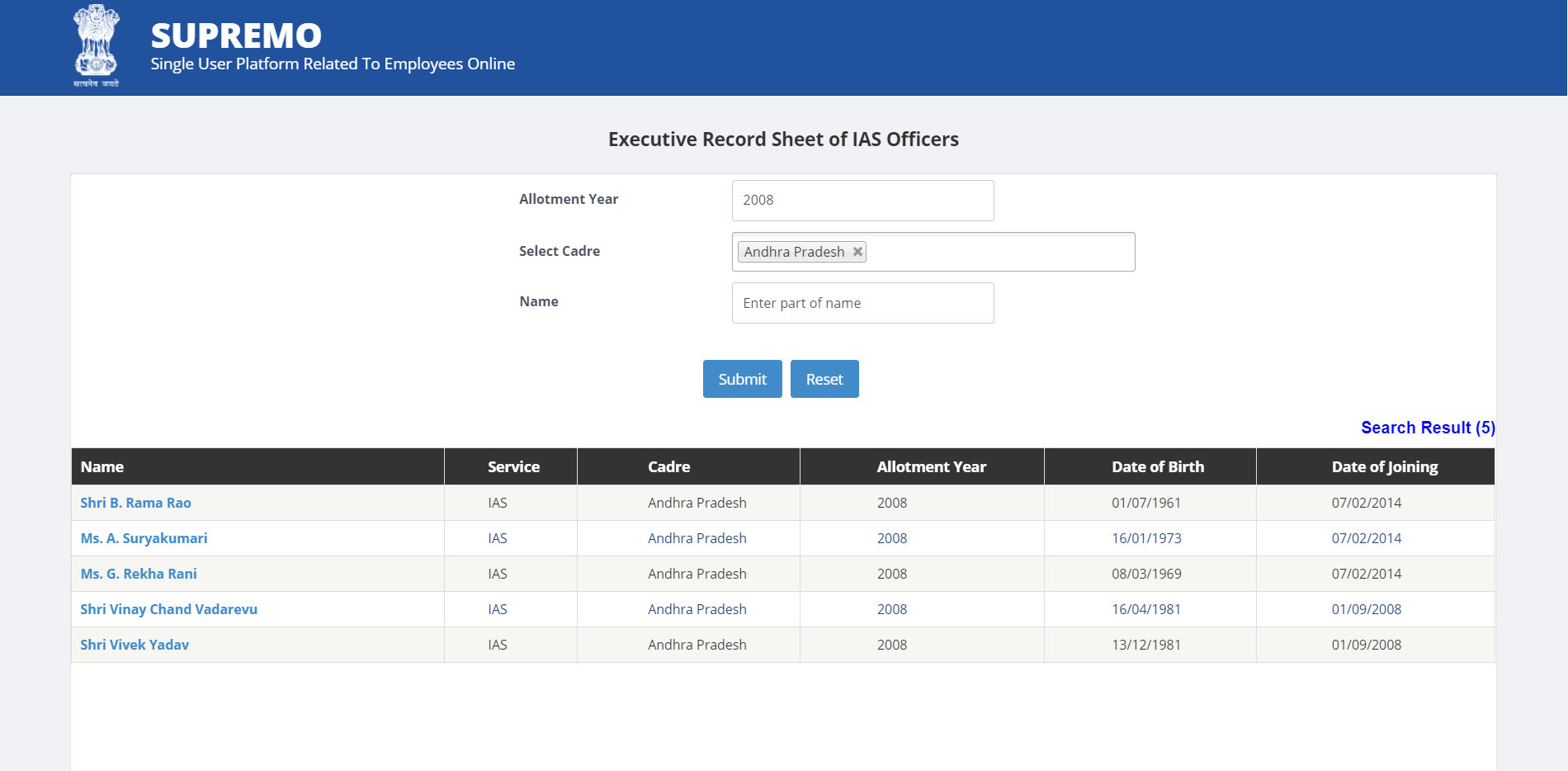
The DOPT website is a relatively simple looking website where the information is dynamically generated in in the form of a table.



Option for cadre can be selected from the drop-down menu, and the website subsequently updates with the relevant information.

1. Supremo (*Hosted at supremo.nic.in)*:

The supremo website is also dynamically generated.



After the relevant options are filled in the “Allotment Year” and “Select Cadre” field, the website generates a table. In this table we can find the “Name” column and click on the hyperlink. Which redirects to this website.



This page contains an in-depth background on a particular Officer (Shri. B. Rama Rao in case of theexample). On further scrolling, we find tables on specifics about the Officers like *Education, Experience, Training history, etc.*

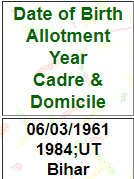
**Methodology in extraction:**

There are 2 parts to the methodology in the scrapping corresponding to the two websites. But the gist of the operation is more or less the same, which is *Table Information Extraction.*

I used Python as it is a versatile tool with un-ending number of open-source modules that makes automation very easy. Here, first tool that can be thought of is [*BeautifulSoup*](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)*,* which is a python package for parsing HTML and XML documents. The problem with *BeautifulSoup* is that it only works with dynamic pages, and [*.asp/.aspx* (Active Server Pages)](https://en.wikipedia.org/wiki/Active_Server_Pages)documents which are being used on the website to generate the relevant information.

I used a tool called [*Selenium*](https://www.selenium.dev/), which is a *portable* framework for testing web applications. Even though the intended purpose of the tool is not to perform scrapping, but it does provide scrapping related tools in the package. It is very to use its python package, which is essentially a wrapper for the tool.

1. DOPT:
2. We can start with spawning a [webdriver](https://www.selenium.dev/documentation/en/webdriver/) (I used [chromdriver](https://chromedriver.chromium.org/)).
3. Then once the driver is loaded, we ask it to get our desired URL.
4. Once the page loads, we can use the pre-defined *XPATH* (XML path language) value to select *<select>* element that corresponds with the ‘Select Cadre’ drop-down box internally through selenium.
5. Now with the selected drop-down element, count the number of *<option>* elements, and remove the first item in the list of ‘options’. The first item in the list is the option for selecting all the cadre values at once, but after testing, it seems that all the entries are not generated even after the first item in the options list is selected, therefore we have to remove the first element.
6. Now we iterate through this list of options, and send a command to click on the options. Once one option is clicked on and the page loads the data, we can perform similar methods to extract information from the table.
7. In the table, some information is concatenated into one cell. Like in this example, each of the lines can be separated into a list format by splitting at ‘\n’. This will form a list object looking like this: [“06/03/1961”, “1984;UT”, “Bihar”]. This list can be mapped to its respective column.



1. Similarly, we can now send a command to select and click the next option in the options list and repeat our table information extraction.
2. Supremo:
3. We can start by spawning 2 webdrivers (I used chromedriver).
4. Once they are loaded, one driver can be asked to get the URL for the website.