Data collection through Webscraping

Rafiq Islam

2024-08-14

Table of contents

## Introduction

Collecting data and preparing it for a project is one of the most important tasks in any data science or machine learning project. There are many sources from where we can collect data for a project, such as

* Connecting to a SQL database server
* Data Source Websites such as [Kaggle](https://www.kaggle.com), [Google Dataset Search](https://datasetsearch.research.google.com), [UCI Machine Learning Repo](https://archive.ics.uci.edu/datasets) etc
* Web Scraping with Beautiful Soup
* Using Python API

## Data Source Websites

Data source websites mainly falls into two categories such as data repositories and data science competitions. There are many such websites.

1. The [UCI Machine Learning Repository](https://archive.ics.uci.edu/datasets)
2. The [Harvard Dataverse](https://dataverse.harvard.edu)
3. The [Mendeley Data Repository](https://data.mendeley.com)
4. The [538](https://github.com/fivethirtyeight/data)
5. The [New Yourk Times](https://github.com/nytimes)
6. The [International Data Analysis Olympiad](https://www.competitionsciences.org/competitions/international-data-analysis-olympiad/)
7. [Kaggle Competition](https://www.kaggle.com)

Example of collecting data from [UCI Machine Learning Repository](https://archive.ics.uci.edu/datasets)

from ucimlrepo import fetch\_ucirepo   
   
# fetch dataset   
iris = fetch\_ucirepo(id=53)   
   
# data (as pandas dataframes)   
X = iris.data.features   
y = iris.data.targets   
   
# metadata   
print(iris.metadata)   
   
# variable information   
print(iris.variables)

{'uci\_id': 53, 'name': 'Iris', 'repository\_url': 'https://archive.ics.uci.edu/dataset/53/iris', 'data\_url': 'https://archive.ics.uci.edu/static/public/53/data.csv', 'abstract': 'A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.\n', 'area': 'Biology', 'tasks': ['Classification'], 'characteristics': ['Tabular'], 'num\_instances': 150, 'num\_features': 4, 'feature\_types': ['Real'], 'demographics': [], 'target\_col': ['class'], 'index\_col': None, 'has\_missing\_values': 'no', 'missing\_values\_symbol': None, 'year\_of\_dataset\_creation': 1936, 'last\_updated': 'Tue Sep 12 2023', 'dataset\_doi': '10.24432/C56C76', 'creators': ['R. A. Fisher'], 'intro\_paper': {'ID': 191, 'type': 'NATIVE', 'title': 'The Iris data set: In search of the source of virginica', 'authors': 'A. Unwin, K. Kleinman', 'venue': 'Significance, 2021', 'year': 2021, 'journal': 'Significance, 2021', 'DOI': '1740-9713.01589', 'URL': 'https://www.semanticscholar.org/paper/4599862ea877863669a6a8e63a3c707a787d5d7e', 'sha': None, 'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': None, 'pmcid': None}, 'additional\_info': {'summary': 'This is one of the earliest datasets used in the literature on classification methods and widely used in statistics and machine learning. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other.\n\nPredicted attribute: class of iris plant.\n\nThis is an exceedingly simple domain.\n\nThis data differs from the data presented in Fishers article (identified by Steve Chadwick, spchadwick@espeedaz.net ). The 35th sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors are in the second and third features. ', 'purpose': 'N/A', 'funded\_by': None, 'instances\_represent': 'Each instance is a plant', 'recommended\_data\_splits': None, 'sensitive\_data': None, 'preprocessing\_description': None, 'variable\_info': None, 'citation': None}}  
 name role type demographic \  
0 sepal length Feature Continuous None   
1 sepal width Feature Continuous None   
2 petal length Feature Continuous None   
3 petal width Feature Continuous None   
4 class Target Categorical None   
  
 description units missing\_values   
0 None cm no   
1 None cm no   
2 None cm no   
3 None cm no   
4 class of iris plant: Iris Setosa, Iris Versico... None no

you may need to install the [UCI Machine Learning Repository](https://archive.ics.uci.edu/datasets) as a package using pip.

pip install ucimlrepo

X.head()

|  | sepal length | sepal width | petal length | petal width |
| --- | --- | --- | --- | --- |
| 0 | 5.1 | 3.5 | 1.4 | 0.2 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 |

## Web Scraping

We scrapping is another way of collecting the data for the research if the data is not available in any repositiory. We can collect the data from a website using a library called BeautifulSoup if the website has permision for other people to collect data from the website.

import bs4 # library for BeautifulSoup  
from bs4 import BeautifulSoup # import the BeautifulSoup object  
import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from seaborn import set\_style  
set\_style("whitegrid")

Now let’s make a html object using BeautifulSoup. Let’s say we have a html website that looks like below

html\_doc="""  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <title>My Dummy HTML Document</title>  
</head>  
<body>  
 <h1>Welcome to My Dummy HTML Document</h1>  
 <p>This is a paragraph in my dummy HTML document.</p>  
 <a href="https://mrislambd.github.io/blog" class="blog" id="blog"> Blog </a>  
 <a href="htpps://mrislambd.github.io/research" class="research" id="research"> Research </a>  
</body>  
</html>  
"""

Now we want to grab information from the dummy html documnet above.

soup=BeautifulSoup(html\_doc, features='html.parser')

Now that we have the object soup we can walk through each elements in this object. For example, if we want to grab the title element,

soup.html.head.title

<title>My Dummy HTML Document</title>

Since the html document has only one title, therefore, we can simply use the following command

soup.title

<title>My Dummy HTML Document</title>

or this command to get the text only

soup.title.text

'My Dummy HTML Document'

This soup object is like a family tree. It has parents, children, greatgrand parents etc.

soup.title.parent

<head>  
<title>My Dummy HTML Document</title>  
</head>

Now to grab an attribute from the soup object we can use

soup.a

<a class="blog" href="https://mrislambd.github.io/blog" id="blog"> Blog </a>

or any particular thing from the attribute

soup.a['class']

['blog']

We can also find multiple attribute of the same kind

soup.findAll('a')

/tmp/ipykernel\_6827/2082462312.py:1: DeprecationWarning:  
  
Call to deprecated method findAll. (Replaced by find\_all) -- Deprecated since version 4.0.0.

[<a class="blog" href="https://mrislambd.github.io/blog" id="blog"> Blog </a>,  
 <a class="research" href="htpps://mrislambd.github.io/research" id="research"> Research </a>]

Then if we want any particular object from all a attribute

soup.findAll('a')[0]['id']

/tmp/ipykernel\_6827/2617565345.py:1: DeprecationWarning:  
  
Call to deprecated method findAll. (Replaced by find\_all) -- Deprecated since version 4.0.0.

'blog'

For any p tag

soup.p.text

'This is a paragraph in my dummy HTML document.'

Similarly, if we want to grab all the hrefs from the a tags

[h['href'] for h in soup.findAll('a')]

/tmp/ipykernel\_6827/1829196164.py:1: DeprecationWarning:  
  
Call to deprecated method findAll. (Replaced by find\_all) -- Deprecated since version 4.0.0.

['https://mrislambd.github.io/blog', 'htpps://mrislambd.github.io/research']

## Example of Webscraping from a real website

In this example we want to obtain some information from [NVIDIA Graduate Fellowship Program](https://research.nvidia.com/graduate-fellowships/archive). Before accessing this website we need to know if we have permision to access their data through webscraping.

import requests  
response = requests.get(url="https://research.nvidia.com/graduate-fellowships/archive")  
response.status\_code

200

The status\_code ensures that we have enough permision to acccess their website data. However, if we obtain status\_code of or then we do not permision or a bad request. For more about the status codes [click here](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status).

soup = BeautifulSoup(response.text, 'html.parser')

We want to make an analysis based on the institution of the past graduate fellows. Insepecting the elements in [this website](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status) we see that the div those have class="archive-group" contains the information of the past graduate fellows.

pf = soup.find\_all("div", class\_="archive-group")

and the first element of this pf contains the information of the graduate fellows in the year of 2021.

pf[0]

<div class="archive-group">  
<h4 class="archive-group\_\_title">2022 Grad Fellows</h4>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Davis Rempe</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Stanford University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Enze Xie (Finalist)</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">University of Hong Kong</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Gokul Swamy (Finalist)</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Carnegie Mellon University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Hao Chen</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">University of Texas at Austin</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Hong-Xing (Koven) Yu (Finalist)</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Stanford University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Mohit Shridhar</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">University of Washington</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Sai Praveen Bangaru</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Massachusetts Institute of Technology</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Shlomi Steinberg</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">University of California, Santa Barbara </div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Sneha D. Goenka</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Stanford University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Suyeon Choi (Finalist)</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Stanford University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Yash Sharma (Finalist)</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">University of Tübingen</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Yufei Ye</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Carnegie Mellon University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Yuke Wang</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">University of California, Santa Barbara </div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Yuntian Deng</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Harvard University</div></div></div>  
<div class="views-row"><div class="views-field views-field-title"><span class="field-content">Zekun Hao</span></div><div class="views-field views-field-field-grad-fellow-institution"><div class="field-content">Cornell University</div></div></div>  
</div>

Now let’s make a pandas dataframe using the information in this page. We can make an use of the output from the above chunk. To grab the year, we see that archive-group\_\_title class with a h4 tag contains the year for all years. With strip=True, the text is cleaned by removing extra whitespace from the beginning and end. We need the first element so a split()[0] will do the job. Then we make another group called fellows that contains the fellows in a certian year by using the div and class"views-row". Once the new group created, we then iterate through this group to extract their names and corresponding institutions.

data=[]  
  
for group in pf:  
 year = group.find(  
 "h4",class\_="archive-group\_\_title"  
 ).get\_text(strip=True).split()[0]  
  
 fellows = group.find\_all("div", class\_="views-row")  
 for fellow in fellows:  
 name = fellow.find(  
 "div", class\_="views-field-title"  
 ).get\_text(strip=True)   
 institute = fellow.find(  
 "div", class\_="views-field-field-grad-fellow-institution"  
 ).get\_text(strip=True)  
  
 data.append({"Name": name, "Year": year, "Institute": institute})  
  
data=pd.DataFrame(data)  
data.head()

|  | Name | Year | Institute |
| --- | --- | --- | --- |
| 0 | Davis Rempe | 2022 | Stanford University |
| 1 | Enze Xie (Finalist) | 2022 | University of Hong Kong |
| 2 | Gokul Swamy (Finalist) | 2022 | Carnegie Mellon University |
| 3 | Hao Chen | 2022 | University of Texas at Austin |
| 4 | Hong-Xing (Koven) Yu (Finalist) | 2022 | Stanford University |

Now let’s perform some Exploratory Data Analysis (EDA). First, we analyze the unique values and distributions.

# Count the number of fellows each year  
year\_counts = data['Year'].value\_counts().sort\_values(ascending=False)  
# Create a DataFrame where years are columns and counts are values in the next row  
year\_data = {  
 'Year': year\_counts.index,  
 'Count': year\_counts.values  
}  
# Create the DataFrame  
year\_data\_counts = pd.DataFrame(year\_data)  
  
# Transpose the DataFrame and reset index to get years as columns  
year\_data\_counts = year\_data\_counts.set\_index('Year').T  
  
# Display the DataFrame  
print(year\_data\_counts)

Year 2018 2022 2019 2017 2015 2006 2012 2011 2016 2007 ... 2021 \  
Count 16 15 15 15 12 12 11 11 11 11 ... 10   
  
Year 2020 2003 2014 2008 2009 2010 2005 2004 2002   
Count 10 10 10 10 10 9 8 7 6   
  
[1 rows x 21 columns]

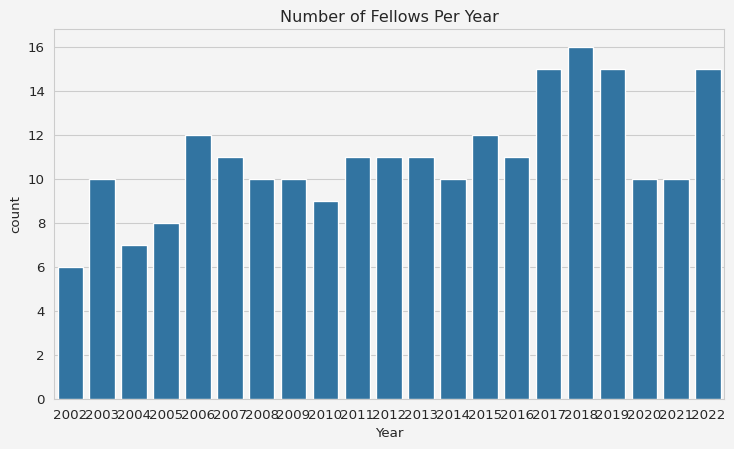
Next we see that most represented universities

university\_counts = data['Institute'].value\_counts()  
print(university\_counts.head(10)) # Display the top 10 universities

Institute  
Stanford University 28  
Massachusetts Institute of Technology 21  
Carnegie Mellon University 17  
University of California, Berkeley 16  
University of Washington 10  
University of Utah 10  
Georgia Institute of Technology 9  
University of Illinois, Urbana-Champaign 9  
University of California, Davis 9  
Cornell University 7  
Name: count, dtype: int64

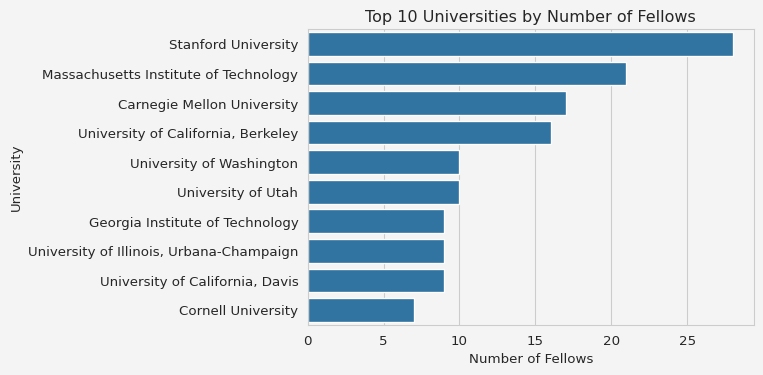
To visualize the award distributions per year,

plt.figure(figsize=(9,5))  
sns.countplot(x='Year', data=data, order=sorted(data['Year'].unique()))  
plt.gca().set\_facecolor('#f4f4f4')   
plt.gcf().patch.set\_facecolor('#f4f4f4')  
plt.title('Number of Fellows Per Year')  
plt.show()



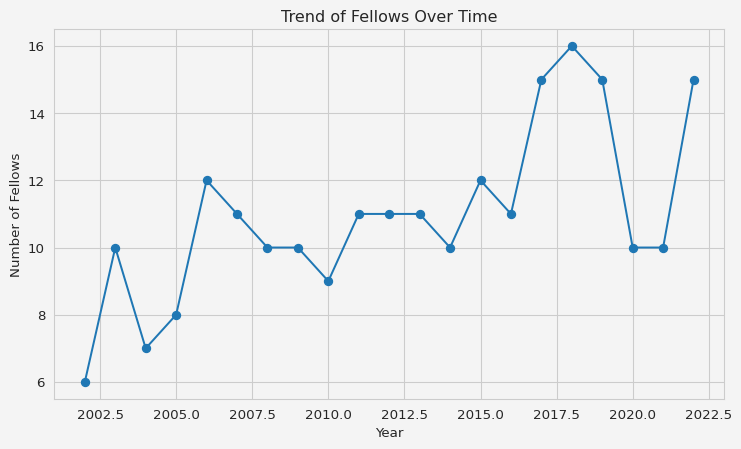
Top 10 universities visualization

plt.figure(figsize=(6,4))  
top\_universities = data['Institute'].value\_counts().head(10)  
sns.barplot(y=top\_universities.index, x=top\_universities.values)  
plt.gca().set\_facecolor('#f4f4f4')   
plt.gcf().patch.set\_facecolor('#f4f4f4')  
plt.title('Top 10 Universities by Number of Fellows')  
plt.xlabel('Number of Fellows')  
plt.ylabel('University')  
plt.show()



Trend over time

plt.figure(figsize=(9,5))  
data['Year'] = data['Year'].astype(int)   
yearly\_trend = data.groupby('Year').size()  
yearly\_trend.plot(kind='line', marker='o')  
plt.gca().set\_facecolor('#f4f4f4')   
plt.gcf().patch.set\_facecolor('#f4f4f4')  
plt.title('Trend of Fellows Over Time')  
plt.xlabel('Year')  
plt.ylabel('Number of Fellows')  
plt.show()



This is just a simple example of collecting data through webscraping. This BeautifulSoup has endless potentials to use in many projects to collect the data that are not publicly available in cleaned or organized form. Thank you for reading.

## References

* [Fisher,R. A.. (1988). Iris. UCI Machine Learning Repository.](https://doi.org/10.24432/C56C76.)

**Share on**

**You may also like**