**Written Deliverable:**

In today’s data driven world web scraping is a common occurrence, the laws surrounding this data gathering process is fuzzy at best. When determining if you should scrape data there are a handful of considerations one must take into account. We as the gatherers of data must understand if we are allowed to gather data from the sources that we choose and understand the legal consequences that we can incur if we do break agreements. Lastly as a data science type, it is important to understand what software and programs are out there to web scrape, and how we can utilize them.

There are some guiding principles that one can adhere to in terms of being a “ethical web scraper.” James Densmore a writer for Toward Data Science, has provided a nice listing of ethical principles that one could adhere to when making the decision to scrape data. First, if you can avoid it do it. Really what this comes down to, is if a website contains an API we should leverage it and avoid scraping all together. Next, when you are scraping you should always provide a User Agent string providing information to your source that will provide them with a way to contact you with any questions or concerns. In addition, you should strive to request data at a reasonable rate so that the source doesn’t think that they’re under a DDoS (Distributed Denial of Service) attack aka a cyber-attack. You should always only take the data that you need and no more, respect the content obtained and don’t pass it off as your own. Another few considerations to take involve adding value for the source, and passing off traffic if possible, responding quickly to inquiries, and again creating value from the data not duplicating it. Here we have it some genuine guidelines that we can adhere to when making the decision to scrape data off the web so we can avoid the negative connotation of being a “bad bot.” (Densemore 2017).

The next question to tackle is how do we know if we can scrape? Thankfully some legislation had been passed in 2016 addressing what is referred to as “bad bots.” Jason Densmore on Towards Data Science put together a comprehensive article addressing current legislation so that we can understand the fundamental principles that were “breached,” and how we can avoid breaking current legislation to have the best approach to obtaining data. Developing a background on this situation, a firm called Bidder’s Edge was indicted by eBay in 2000; they were accused of trespass to chattels which in plain English means that Bidder’s Edge interfered with the eBay’s personal property. In this scenario, Bidder’s Edge was indicted because they scraped eBay’s pricing in order to be competitive. The reason why the court granted injunction is because users are required to agree with eBay’s terms of service agreement, and that the bots were disruptive to eBay’s systems. This was the start of the BOTS act passed in 2016, this act targeted the bots that drove up the prices of tickets on Ticketmaster and other ticket vendors. Specifically, it enforces purchasing limits for tickets greater than 200 persons, prohibits sales of tickets where this deceptive buying practices took place. (Roberts 2018). So now it is important to understand how we can avoid such legislation in web scraping practices, and the fundamental principle that you should read and understand the source’s terms of agreement; violation of these terms could result in a big fine or legal repercussions. So at the very basic level, you should obtain written permission to scrape a website and read the terms of agreement to ensure that you’re not in violation. Legal consequence of breaking these regulations can result in fines, felony convictions, and even jail time, so it’s not something to take lightly.

At this point it has become very apparent what we shouldn’t do, in an effort to avoid legal ramifications. But that is not exciting, what is exciting, involves the various technologies that we can leverage to web scrape. To make things easier Jacob Koshy on Prompt Cloud published an article that outlined some of the various technologies that are out in the tech space that we can leverage to accomplish data collaboration via web scraping. The first of many outlined was Selenium; a browser automation tool that allows users to utilize a broad range of tasks on autopilot. It allows the user or developer to access websites just as a user would; with tools to automate and perform testing; learning this too will offer a great deal of flexibility to users. Boiler Pipe is another option, a Java library that does a great job of extracting text an titles in a clean and orderly fashion, removing undesired html tags and other noise in just nanoseconds. Nutch, an open source web crawling tool has become “the gold standard,” of web crawling offerings, under the hood is a very complicated crawling algorithm that makes it one of the best tools to leverage in terms of scraping. Another notable tool is Ruby’s Waitr (pronounced water) its an easy to use and configure program that can interact with websites just as a person would, by clicking into links, filling out forms, and so on.(Koshy 2019) These four programs just scrape the surface of what is available out in the web-scraping universe.

Densemore, Jason. Ethics in Web Scraping. July 23, 2017. <https://towardsdatascience.com/ethics-in-web-scraping-b96b18136f01>

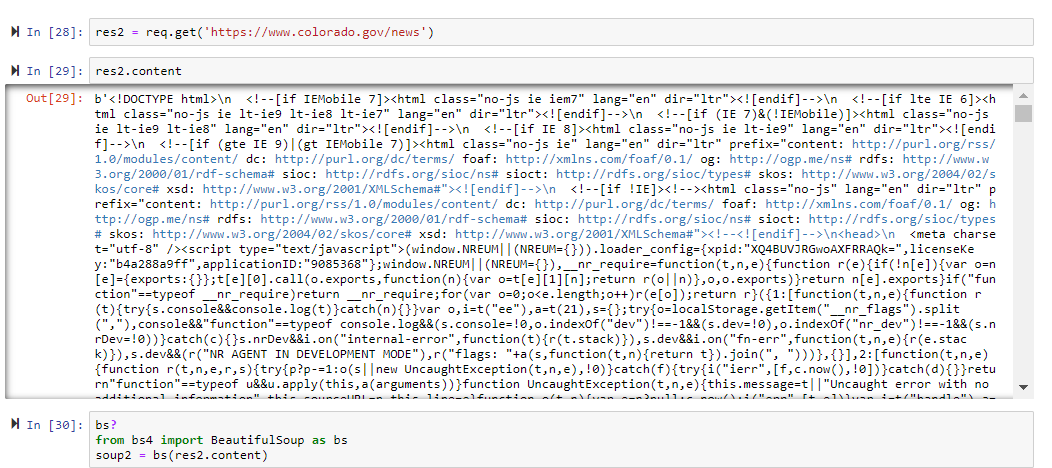
Roberts, Edward. Is Web Scraping Illegal? Depends of What the meaning of Word is. September 17, 2018. <https://www.imperva.com/blog/is-web-scraping-illegal/>

Koshy, Jacob.5 Technologies to Master if you Want to Scrape the Web August 19, 2019 <https://www.promptcloud.com/blog/technologies-for-web-scraping/>

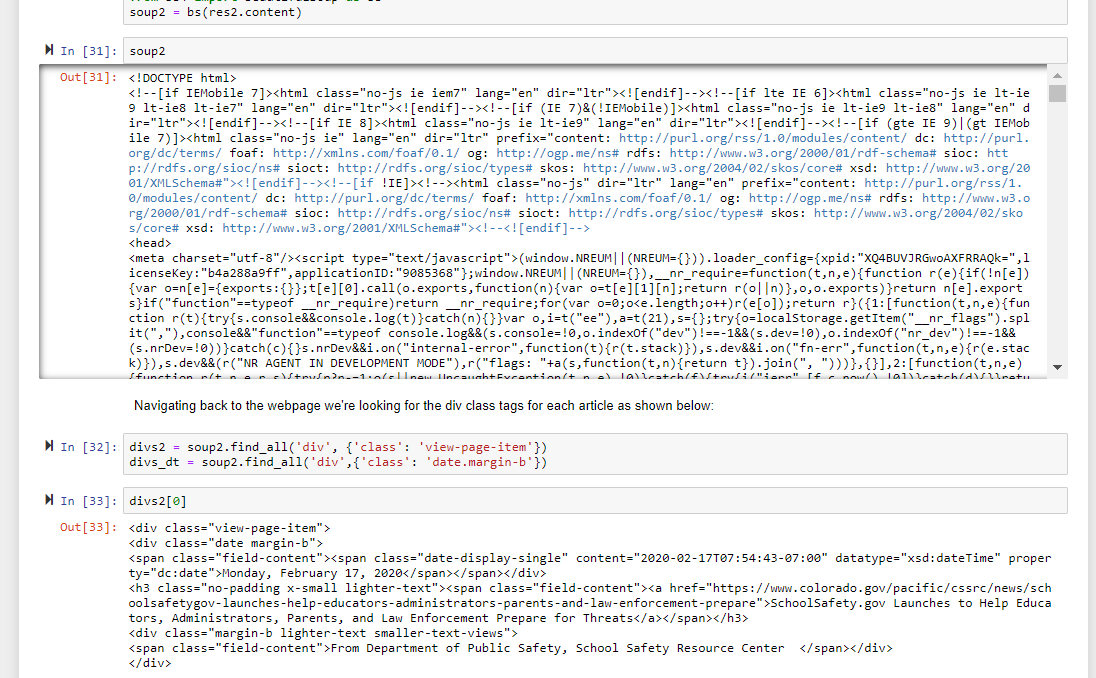
**Technical portion:**

For my deliverable this week, after completing the example on the Denver News website. I then decided to scrape the Colorado News website.

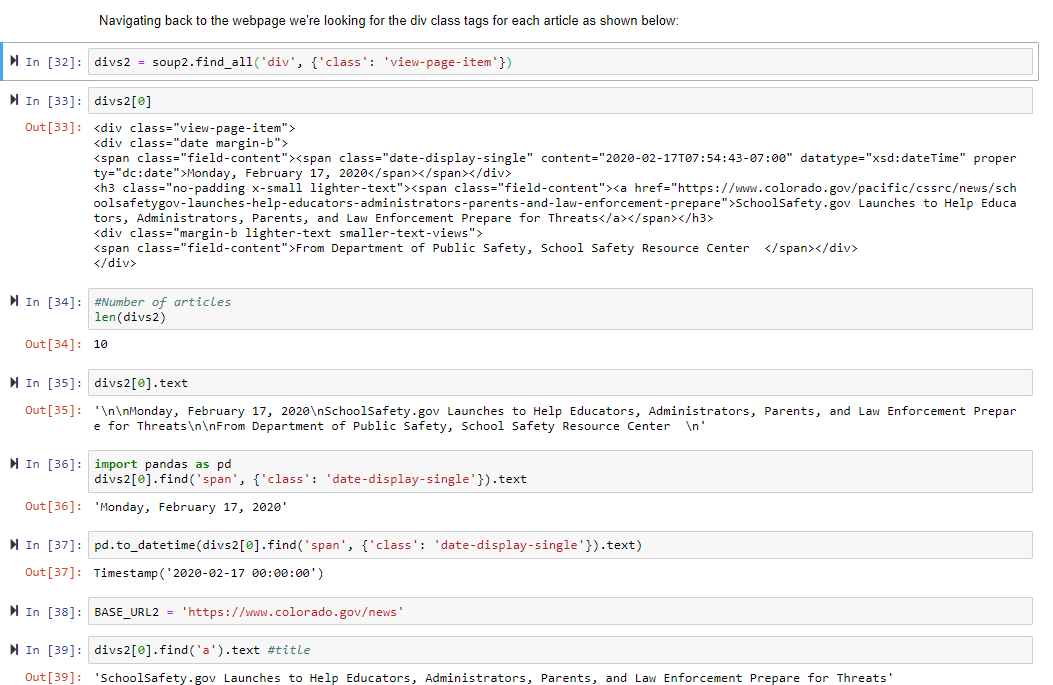
First I started by grabbing the raw content from the website as shown below:



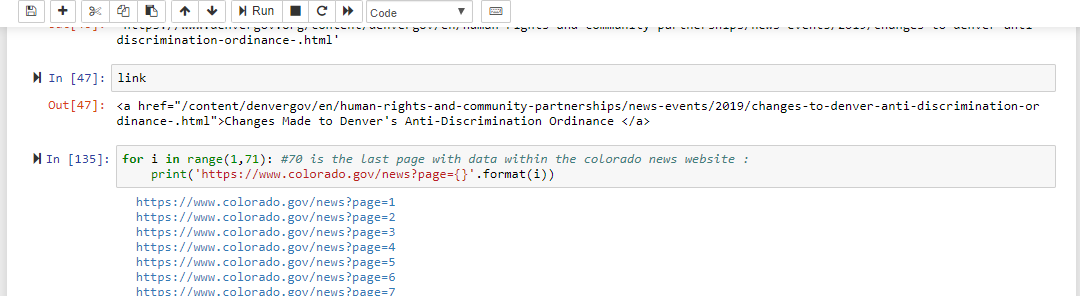
Then I utilized beautiful soup to parse out the data shown in soup2 below:

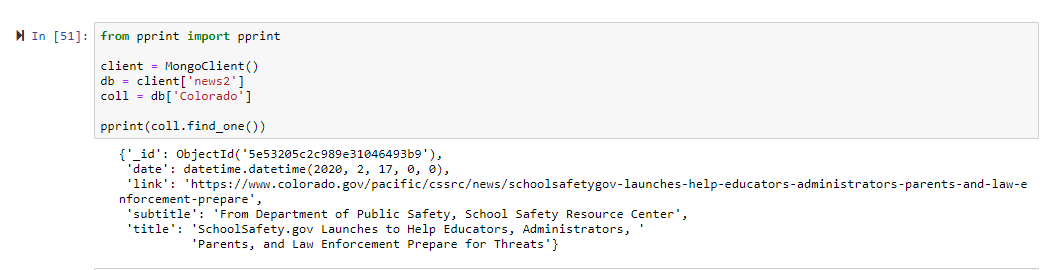


By referencing the divs on the webpage I was able to parse out the articles as shown below, getting a count of articles, parsing out the date, title, and other relevant data so that we can eventually get this into MongoDB.









Now going back to MongoDB we can see that the extraction was successful:

