MassCourts Scraper Project Report

Team Members: Frank Pacini, Will Regan, Justin Sayah

Client: Massachusetts Joint Committee on Housing

Contact: Taylor Trenchard

## **Project Goal**

With the rise and fall of Covid cases as well as the institution and removal of federal and local eviction moratoriums, the general status of housing has been unstable across the USA. Under guidance of the Massachusetts Joint Committee on Housing, the overall goal of this project is to employ data analytics to develop a better understanding of eviction trends and patterns over the last two years in order to help inform the formation of public housing policy for Massachusetts.

Trends we hope to identify include those relating to geography, race, ethnicity, family composition, time periods, income, or other unidentified trend vectors. We also hope to identify how federal and local policies affected these trends and continue to affect them.

To achieve the above goal, this team was assigned to scraping public Massachusetts eviction data from the [Masscourts](https://masscourts.org/) site in order to expand the amount of data available for the other groups of our project to use for analysis. Therefore, our group goal was to create a functioning codebase that would be capable of scraping all desired fields of relevant court cases within desired time periods from the [Masscourts](https://masscourts.org/)site reliably.

## **Summary**

Over the course of the semester we built a web scraper for the MassCourts site which specifically targets Massachusetts eviction (summary process) cases in the housing and district court divisions. The scraper is written in Python using the Selenium library for site navigation and data retrieval, as well as the ProxyScrape API to obtain IP proxies for anti-detection. We run our code in Jupyter notebooks to separate functionality and for ease of use. Using the scraper, we’ve obtained a dataset of eviction cases for a month prior to the eviction moratorium (January 2020) and a month after the moratorium ended (January 2021) for all housing courts and selected district courts with high per capita eviction rates. The dataset includes many fields taken directly from the case documents, as well as ones we deemed as relevant and extracted from the document text after extensive qualitative analysis of site eviction cases. We also performed preliminary analysis of this extended dataset and the new extracted fields.

## **Scraper Construction & Functionality**

Three goals primarily informed our design of the scraper:

1. Collect as much data as possible and parse out all relevant fields
2. Run as quickly as possible
3. Avoid being detected and blocked by the site

The most efficient way to accomplish (1) and (2) would be to use a webdriver to open a browser to Masscourts and automatically navigate through cases using the site’s forms, collecting all the data from each document as we go. Using the python package Selenium allowed all of this to be accomplished with one library.

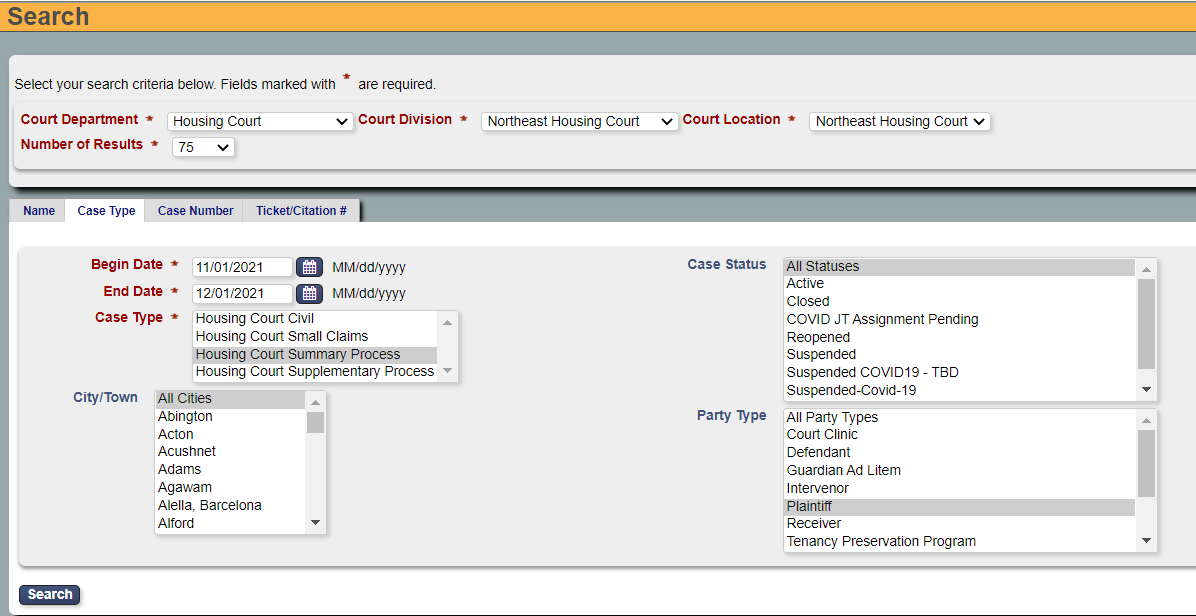
Since this method makes many page requests in quick succession, though, (3) necessitated finding a way to run the scraper without being blocked by the site for overuse. One way to accomplish this is to run the browser under a proxy IP so that Masscourts thinks the requests are coming from somewhere else. If any IP is detected and banned, we can simply rotate to the next IP address and continue. To do this, we used the free API for ProxyScrape.com, which compiles an up to date list of free proxies available for use. In our code we request this list and then run through each one in a random order until we find one which is sufficiently fast for the scraper to work.

The next hurdle was getting past the site’s CAPTCHA, which it uses to prevent entry into the site by bots:



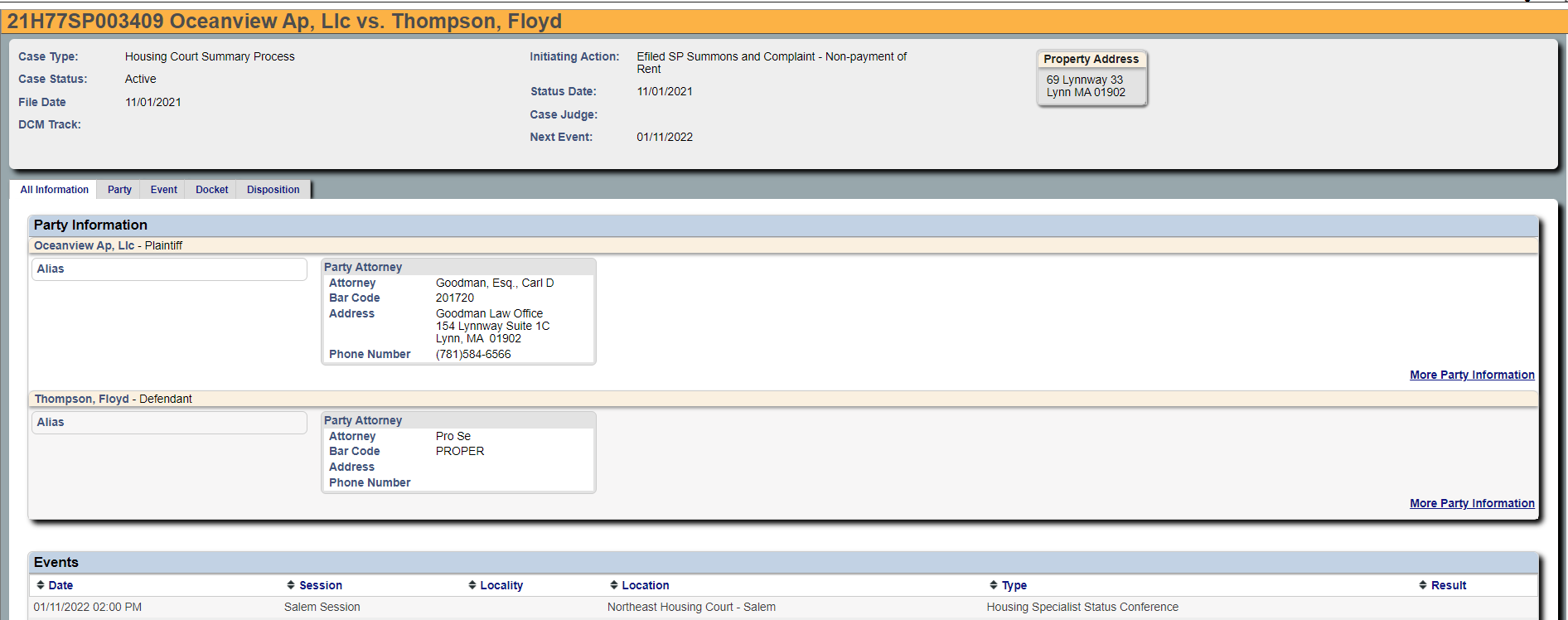
We decided to use the DeathByCaptcha service for this, which we can query in our code to solve the captcha for us. Although an automatic method is necessary to keep the scraper efficient and avoid user input, for testing purposes we decided it is better to just solve the captcha manually before running the next section of the code, since the service is paid.

The next page is the form to search for case documents:



At this stage we began putting random wait times while navigating the site in order to look more human-like, in case the site is using any services which detect unusual activity. We also began handling the Housing and District courts separately. To complete the form, we used Selenium to select the desired form elements, enter our date query and submit. The search is separated into court divisions, so we needed to gather all data one at a time for each housing court. We changed the number of results from 25 to 75 since this more often avoids having multiple pages of results to navigate. For dates, we can only choose date ranges up to a month and so had to split up any larger queries. We focused our data curation only on Summary Process cases so chose that as the case type. Case results are also duplicated for each party in the case, so we found it helpful to filter party type to plaintiff, since it is less likely that there are multiple landlords than multiple tenants in any one case.



On the search results page, we found that the number of results displayed for a query is capped at 100. Because of this, it was necessary to split up any desired date range into single day queries. So for each housing court division, we make a query for each day in the desired range, grab all the data from the query results and then return to the search. The results can still include duplicate cases, so we must skip over duplicate numbers when iterating through each case link. For efficiency, we grab the links all together and then request each page one at a time.

Each case document contains a header with basic case information, information about each party in the case, a list of events that have occurred or are scheduled, a docket which contains detailed information about the case proceedings, and a section for case results. Our goal was to grab all information and include it in the dataset, but also parse out more specific fields that could be relevant to case analysis. After an extensive review of different case proceedings and background research, we decided on the following fields for the dataset:

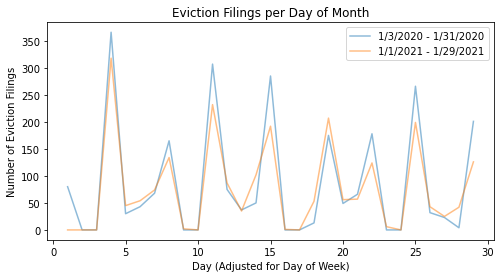
| **Field Name(s)** | **Explanation** | **Reason for Inclusion** |
| --- | --- | --- |
| caseNum | The unique id for this eviction case | Can search for case to manual examine or for updates and remove duplicates |
| courtDepartment | The court department, in our case either Housing or District | Can compare district and housing court trends |
| courtDivision | The division of the court, split regionally for Housing and municipality for District | Can compare eviction filings between housing courts |
| caseType | Always “Summary Process” | Included for completeness |
| caseStatus | Status of the case. Either active / open or closed for Housing, plus transferred or disposed for District | Should be updated later if active, if transferred, should consider housing court case |
| fileDate | When the eviction action was initiated | Eviction trends over time |
| fileType | How the initiating action was filed (e-filing or not). Also says whether the cause was non-payment of rent or something else | How people are filing evictions, trends in eviction reason |
| address | The address where the defendant / tenant lives | Evictions by city / region |
| result | The result type for this case | How many actual evictions occurred, trends in case results |
| resultDate | The date when the case finished | Time between filing and result date, and when the tenant was actually evicted |
| plaintiffs | A list of plaintiffs / landlords separated by “||” | Are certain landlords evicting more people? |
| defendants | A list of defendants / tenants separated by “||” | Completeness |
| plaintiffAttorney  plaintiffAttorneyCode  plaintiffAttorneyAddress  plaintiffAttorneyPhone | The name, license number, address and phone number of the first attorney assigned to the plaintiff(s). | Record of specific attorneys in eviction cases |
| defendantAttorney  defendantAttorneyCode  defendantAttorneyAddress  defendantAttorneyPhone | The name, license number, address and phone number of the first attorney assigned to the defendant(s). | Are tenants being represented and by who? |
| eventDates  eventSessions  eventLocations  eventTypes | Encoding of the Event section. The date, session, location and type (i.e. Trial) of each event, separated by “||” | Important characterization of case proceedings included for further processing |
| nextEventType  nextEventDate | The most type and date for the most recent listed event, regardless of whether it has occurred yet | Could analyze the event that led to result, or track the scheduled event for active cases |
| hadStatusConference  hadMediation  hadTrial | Boolean values for whether each event type has occurred in this case | Based on analysis of different cases, these were a significant indicator of how the case played out |
| judge | The name of the presiding judge if the case went to trial | Track rulings of different judges |
| amount | The amount the tenant was ordered to pay if the court ruled in favor of the plaintiff | How much are tenants forced to pay or how far are they behind on rent, etc. Also confirms that the landlord won the case |
| caseDismissalReason | The reason listed if the court dismissed the case | Additional information on a result type |
| answerDate | The date of answer and counterclaim of the defendant if it occurred | Significant event which occurs only in certain cases and often leads to an agreement for judgement. Could track results when defendant responds |
| docketText | Encoding of the case docket entries. List of docket entries separated by “||”, and the docket entry date and text for that entry are separated by “\_” | All available information to characterize case proceedings. Included for further processing |

Besides the basic information included in all case documents, the additional fields we’ve extracted from the docket and event text (in bold) are primarily based on useful information included in the text under specific result types. For instance, the amount paid is only included if the plaintiff won, in which case the result is always “Judgment in SP by Default”, “Judgment in SP case entered”, or “Findings and Order”. It is still possible in the latter two types for the defendant to win, so the amount field confirms who won the case while also being valuable information in itself.

## **Dataset Overview and Preliminary Analysis**

The dataset we’ve compiled using the scraper contains all 4,755 eviction cases opened in Jan 2020 and Jan 2021 either in the Massachusetts housing courts, or in the district courts for gateway cities Springfield, Brockton, Lowell, New Bedford and Worcester.

Below is a collection of graphs produced with the scraped data found in the [ExampleDatasets Folder.](https://drive.google.com/drive/folders/1EOyJ0TYO9SpK47Duf8RZqRTg1QPvSsdu?usp=sharing) These graphs show the proportion of certain case statuses, results, etc. in our example dataset.

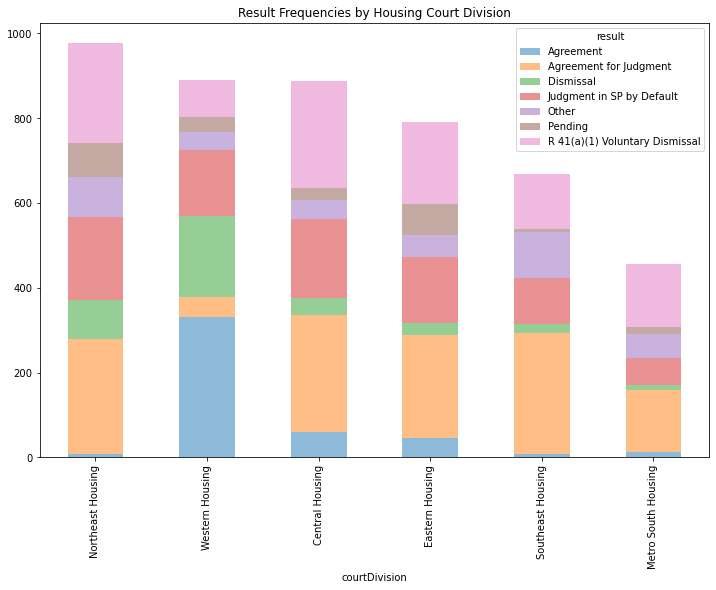
**Eviction Filings per Day of Month**

This graph displays the amount of Eviction Filings on a per day basis, adjusted to start on the same day of the week. We see that overall the spikes are similar between the two January months, with 2020 usually having a bit more evictions during spikes.

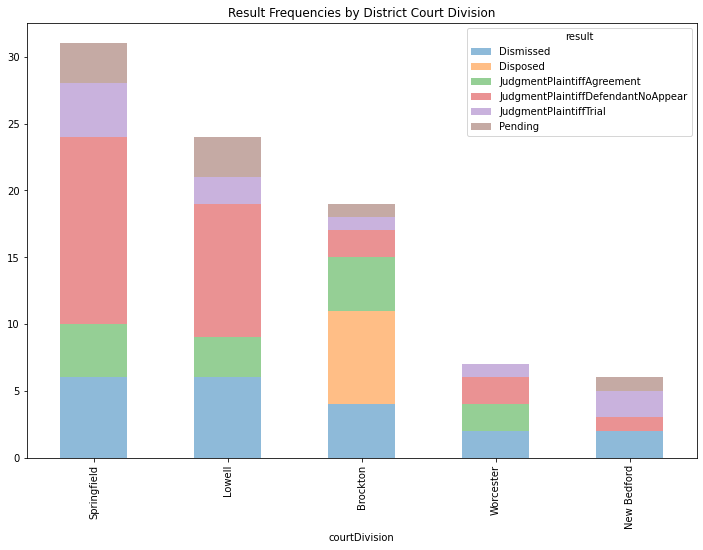
**Most Common Result Types**

| Result | Number of Cases 2020 | Number of Cases 2021 |
| --- | --- | --- |
| Agreement for Judgment | 1001 | 268 |
| R 41(a)(1) Voluntary Dismissal | 372 | 672 |
| Judgment in SP by Default | 570 | 295 |
| Agreement | 233 | 234 |
| Dismissal | 102 | 277 |
| Pending | 66 | 184 |
| Disposed | 74 | 105 |
| Stipulation of Dismissal | 41 | 100 |
| Total of All Results | 2539 | 2214 |

This table displays the distribution of most common result types during the sampled time periods, compared by year. We can see that 2021 had far fewer results of Agreement for Judgement and many more of R 41(a)(1) Voluntary Dismissal.

**Result Frequencies by Housing Court Division**

### **Result Frequencies by District Court Division**



The above two graphs show the distribution of case results per housing and district court division. We can see that the western housing court has many more “Agreement” and “Dismissal” results than the others. Brockton is also the only district court to have the “Disposed” ruling, and the two larger districts have larger proportions of no appearance and pending cases than the smaller three.

**Most Frequent Plaintiff Attorneys**

| Plaintiff Attorney | Number of Cases |
| --- | --- |
| Pro Se | 599 |
| Scolnick, Esq., Stephen M | 351 |
| Turk, Esq., Jeffrey Craig | 267 |
| Flynn, Esq., Frank Anthony | 179 |
| Higgins-Shea, Esq., Katharine Ann | 103 |
| Van Dyke, Esq., James Howard | 88 |
| Gouveia, Esq., Lisa M | 87 |

This table shows some of the most frequent plaintiff attorneys. We can see that there are some attorneys that are heavily represented. However, we can see that approximately 10% of plaintiffs represented themselves. It may be worth analyzing common attorney patterns and records.

**Most Frequent Defendant Attorneys**

| Plaintiff Attorney | Number of Cases |
| --- | --- |
| Pro Se | 3976 |
| Ahern, Esq., Sean M | 11 |
| Bhatti, Esq., Mariyam B | 8 |
| Mount, Esq., Robert John | 7 |
| Robin, Esq., Jeremy Todd | 7 |
| Alfred, Esq., Elizabeth P | 7 |
| Pisegna, Esq., Vincent J | 6 |

This table shows the most frequent defendant attorneys. We can see that tenants are very rarely represented by an attorney. Based on our investigation, several of the most frequent attorneys here work for pro bono organizations Community Legal Aid and Northeast Legal Aid.

**Most Frequent Addresses**

| Address | Number of Cases |
| --- | --- |
| 105 Spruce Street Watertown MA 02472 | 3 |
| 342 Middle Street East Weymouth MA 02189 | 3 |
| 47 Spruce Street 4 Clinton MA 01510 | 3 |
| 364 Belmont Avenue 26 Springfield MA 01108 | 3 |
| 174 Center Street 2nd fl Ludlow MA 01056 | 3 |
| 21 Niagara Street 4R Springfield MA 01105 | 2 |
| 302 Hollis Street 1 Framingham MA 01702 | 2 |

Although none of the addresses go above 3 cases, which is still numerous for a two month period. It may be worth looking if the most common eviction locations have some sort of similar patterns such as area demographics.

**Most Frequent Judges**

| Address | Number of Cases |
| --- | --- |
| Michael Neville | 141 |
| Nickolas W. Moudios | 100 |
| Hon. Gustavo A Del Puerto | 75 |
| John M. Goggins | 67 |
| Hon. Donna Salvidio | 64 |
| Hon. Joseph Michaud | 43 |
| Ho. Fairlie A Dalton | 42 |

This simple table displays the most frequent judges on cases for both 2020 and 2021. We can see that there are a few judges that have many cases, but 141 is still under 4% of the total cases. It might be worth the time to analyze popular judges’ decision patterns.

### **Amount Owed Frequencies**

The “Amount Owed Frequencies” graph is a histogram of the amounts owed by the defendant if found guilty. We can see a very clear downward, curved trend that suggests evictions happen much more in lower cost housing units. There are still a few amounts for each higher bracket, and these tend to be businesses that would have higher rent payments.

**Most Frequent Case Dismissal Reason**

| Reason | Number of Cases |
| --- | --- |
| “Plaintiff’s request” | 73 |
| “Neither party appeared” | 28 |
| “By order of the court” | 17 |
| “Plaintiff failed to appear” | 11 |
| “Neither party appeared. This case will be dismissed in seven days if no action is taken.” | 7 |
| “Plaintiff's request. Voluntary dismissal filed” | 5 |
| “Judgement of Dismissal” | 4 |
| “Judgment of Dismissal. Lack of Jurisdiction” | 2 |

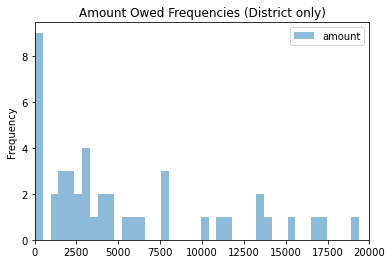
We can see from the Dismissal frequencies table how often each reason for dismissal occurred when a case was dismissed. In the table, we see “Plaintiff’s request” being the most common reason, which might mean the landlord might give the defendant more time to pay. We also see that “Neither party appeared” to be common, which suggests either both sides forgot to show or that they did not go because some other arranged. “By order of the court” was also frequent and means the judge might have decided the defendant deserves more time or that the case was not made properly.

**Most Frequent Next Events**

| Next Event Type | Total Number of Cases | Number of Cases 2020 | Number of Cases 2021 |
| --- | --- | --- | --- |
| Summary Process Trial | 2315 | 1745 | 570 |
| Motion Hearing | 779 | 497 | 282 |
| Housing Specialist Status Conference | 531 | 41 | 490 |
| Status Hearing | 330 | 28 | 302 |
| Housing Specialist Mediation | 298 | 41 | 257 |
| Review | 121 | 54 | 67 |
| Trial | 97 | 13 | 84 |
| Jury Trial | 41 | 40 | 1 |

This table compares the types of upcoming or most recent events for each case. We can see a difference in the distribution as 2021 has much less Summary Process Trial, less Motion Hearings, and much more in Housing Specialist Status Conference, Status Hearing, and Housing Specialist Mediation.

### **Amount Owed Frequencies (District Only)**

****

Similar to the previous Amount Owed Frequencies graph but for districts, this graph shows the starting of a similar trend as observed in the previous graph. The most common amount seems to be in the low $100 range. We see the most cluster around $2500. This cluster is approximately surrounding the costs of rent in Boston, so we can assume this cluster is late rent payments of individuals, families, or groups.

## **Technical Challenges**

In the making of our scraper, we faced 3 major obstacles:

1. The construction of the Mass Courts site and site updates
2. The time it takes to scrape the site
3. Issues with free IP services that cause frequent errors

The construction of the Mass Courts site made it tricky to scrape at times. To find eviction data over a specific date range, the scraper must check certain boxes such as which specific court to scrape and specifically request Summary Process trials. When selecting secondary boxes like “Plaintiff”, our scraper would go too fast for the site to update our selections accordingly, so we had to manually slow down our scraper at times to accommodate the site, which only adds to the second major obstacle listed above. Furthermore, the site splits the housing and district courts into divisions. For housing, there are six divisions and sixty-two districts. Since our goal was only getting evictions over a certain time range, we wanted to go through all divisions, which caused our scraper to take six times longer for housing and many times longer for districts than if all the cases could be listed on one page. Furthermore, one of the most problematic intricacies was date selection. The Mass Courts site does not allow date selections more than 30 days apart, which made it so at best we could only go through the data month by month. However, even worse, the site does not display cases after 100 cases are already reported in a search timespan. This means that we had to knowingly select time spans that had less than 100 cases, which happened to be only 1 day increments. Even this strategy of one day at a time was not enough as some days have more than 100 cases. Finally, the organization of the case information itself can be very convoluted and non-standardized, which makes the process of categorizing as we scrape difficult.

While building our scraper, changes were made to the site, forcing us to make modifications to our scraper. Near the beginning of the semester, they changed their Captcha to enter the site, which made most of our work of the first week useless, but it also was an easier CAPTCHA to crack with Death By Captcha. One of the more recent changes was that sometimes the site encounters Internal Server errors. These errors did not happen for much of our project until near the end. These happened usually in the midst of heavy scraping, which made us believe they are trying to implement more anti-scraping methods.

As mentioned above, the time it takes to scrape the site is massive because of the need to go through each district and each day individually. This means that one day of results must go through six different selection screens, wait for the response from the server (which can take a while on our IPs), then scrape each case on each response. We also put in some brief breaks to avoid spamming their server and/or getting our IPs banned, which only slightly added to our runtime. The biggest issue with time is not the waiting, but the fact that the longer the scraper goes continuously, the more chances the IP could give out or Selenium could encounter a bug. Each day takes around 5-10 minutes (usually on the higher end) to scrape depending on the case volume. Again, this is not a big issue by itself, but it is compounded with the next obstacle.

To avoid getting our local IPs banned, we used free Proxy IPs. These free Proxy IPs are mostly slow and unreliable, but they are free, which makes the cost of them getting banned very low. The slowness of these IPs only compounded on the earlier mentioned issues of time. The biggest obstacle of these IPs was the unreliability, however. Often, after a few minutes of scraping, the IPs would disconnect or become unresponsive, making our Selenium program timeout or reach an error at any line in its execution. Because these errors could happen anywhere in execution, catching and recovering from them becomes very difficult. Furthermore, sometimes an IP issue could be masqueraded as a TimeoutError, which is the same error given by Selenium if it fails to find a certain element, which also happens often. Sometimes, you could fix the renderer issues by refreshing the page a few times, but this fix seldomly worked. Overall, this obstacle makes our scraper incredibly inconsistent as the scraper can run fine for an hour and possibly get a whole week with no issues, but then it could also fail on the first day. Our luck was very dependent on the quality of free IPs we could find and use.

## **Moving Forward**

This project is unique because it is continuing into the Spring with a more professional team and with different tools. The tools they are using in the Spring address many of the issues we have run into during our project. We will go through each issue presented in the previous section and discuss how that is being resolved in the future iteration of this project.

Firstly, the issue of the construction of the site is approached by not categorizing the data as it goes. The approach of the new scraper will be to collect everything first, then sort out what was collected. This process will be done by going to each case and simply copying and saving the page’s HTML instead of parsing through the page looking for each field in each case. This approach splits the issue of the incongruencies in case files from the scraping part. Our scraper could fail to notice some details in outlier case files and either throw an error or miss certain data, which means we would need a rerun. However, collecting all the raw HTML means that the outliers can be addressed after scraping the site.

The new scraper will use a paid IP service for faster and more reliable IPs. This will greatly reduce the time it takes to scrape as the quality will be higher, and as a byproduct it also reduces the chances of renderer and IP errors. The new scraper will mainly address the issue of time by adding more parallel running scraping instances. On our local machines, each person could run 1-3 instances of the scraper realistically. The next project will use Amazon services to house 20 or more parallel processes, which may not directly increase the time of each individual scraper, but can allow the collection of mass data in much less time as each instance handles its own dates. Furthermore, the new scraper can more easily replace one of the instances if it fails.

Although our scraper is not going to be utilized in the next scraper, some ideas, techniques, and experiences that we learned are going to be useful. For example, we already have a working Death By Captcha service for the current captcha. We have also done extensive work on extracting data and features from the Docket Text of each case, which can be utilized after the data is collected in the new scraper. The project after ours is using a scraper that worked on an old version of the site, so if they consult the scraper we have now, they can have a framework on how to approach newer selections and site intricacies like server error recoveries.

## **Appendix**

**Links to Scraper and Dataset**

[Scraper Repository](https://github.com/BU-Spark/ds-state-ma-housing-comitt/tree/group2-scraper)

[Project Folder (with scraped datasets)](https://drive.google.com/drive/folders/1EOyJ0TYO9SpK47Duf8RZqRTg1QPvSsdu?usp=sharing)

**Walkthrough of Methods and Code:**

The first cell of our scraper includes necessary libraries and must be run before any other cell.

The second cell of our scraper connects us to a free proxy IP and starts our chrome webdriver. It will rotate to the next IP if the old one does not respond in time.

* do\_dbc(driver): This method takes the driver as input and uses the Death By Captcha Service to solve the Captcha. This method currently is not employed in the scraper, but it does work. We do not use it as Captchas only need to be solved once.
* sleep\_rand(): This method sleeps the python execution for a random time between 1 and 2 seconds (uniform) to avoid bot detection.
* choose\_court\_division (driver, court\_division): This method chooses Housing court and the inputted court division for the Housing Court Scraper. It also sets the number of results per page to 75
* type\_dates(driver, startdate, enddate): This method types in the selected start date and end date into the selection window. If no end date is specified, the end date is set to be the start date.
* clear\_dates(driver): This method clears the inputted dates. Used after coming back to selection window and before putting in new dates
* select\_summary\_process\_and\_plaintiff(driver, selection): This method selects to search Summary Process courts and only get Plaintiff perspective case listings.
* get\_event\_fields(party): This method is part of the case extraction suite. This method gets information on each event the court reported.
* get\_attorney\_fields(party): This method is part of the case extraction suite. This method gets the Attorney information of each party if there is any.
* get\_party\_fields(driver): This method is part of the case extraction method. This method gets the plaintiffs and defendants information along with their attorney information.
* extract\_data\_from\_doc(driver, court\_department, court\_division): This method extracts data from a case file given the webdriver, which department the case is in, and division. This method returns a complete list of all desired features of the specified case.
* get\_date\_list(start\_datetime, end\_datetime): This method returns a list of date strings from the inputted start datetime python object to the end datetime python object. The end date is set to the current date if no end date is entered. The returned strings are in order and can be directly inputted into the date selection box on the site.
* get\_cases\_list(driver): This method gets a list of each case number on a specific search result.
* get\_case\_links(driver): This method returns a list of the url links to each case after searching from the selection screen.
* save\_dataset(dataset, startDate, endDate, court): Given a list of case information lists (list of extractions), this saves a dataset in the data folder named with its court, start date, and end date in the form of a csv.
* Housing Court Main Runner: This section of the notebook contains the main runner for our Housing Court scraper. This uses each method above to collect and save scraped data into a csv. This method will return an error if the IP dies or renderer stops responding.
* District Court Side: This section contains methods for the district scraper specifically and the main District Court Scraper.
* choose\_court\_division\_district(driver, court\_division): This method is the district version of the similarly named housing method that selects District Court instead of Housing court.
* find\_status(text): Given the docket text of a district court listing, this method returns the case status/result.
* District Court Main Runner: This section of the notebook contains the main runner for our District Court scraper. This uses all of the methods above to collect and save scraped data into a csv. This method will return an error if the IP dies or renderer stops responding.