

## **Background**

Mulberry is a start-up company that connects businesses, insurance companies and customers all together by enabling e-commerce retailers to offer extended protection plans at the point-of-sale. When customers shop on merchants that have integrated Mulberry plugin, they can simply add corresponding warranties along with their purchases as part of their normal checkout process. However, it's impossible that every type of product is warrantable. In order to make this process work, our group needs to build an end-to-end classification system that takes a product title, description, and price, and returns the product category. For example, it would take Samsung 42" 4K HDTV 10-bit Adobe HDR as a product title, a description of the product, and \$749.99 as product price, and return "electronics/televisions" as a classification. Returning an accurate product category using Neural Network is crucial for our project so that the company can use the result to properly price the warranty instantly.

## **Data**

During our first meeting, our sponsor showed us the data they used to train their initial models. Basically for every product, it has several columns that contain the information they scraped from the online retailers (e.g. price, product title, description), the classification it falls into, and its category from the insurer's perspective. Before sharing these data with us, they would like us to first get started on scraping some websites on our own to collect some new data.

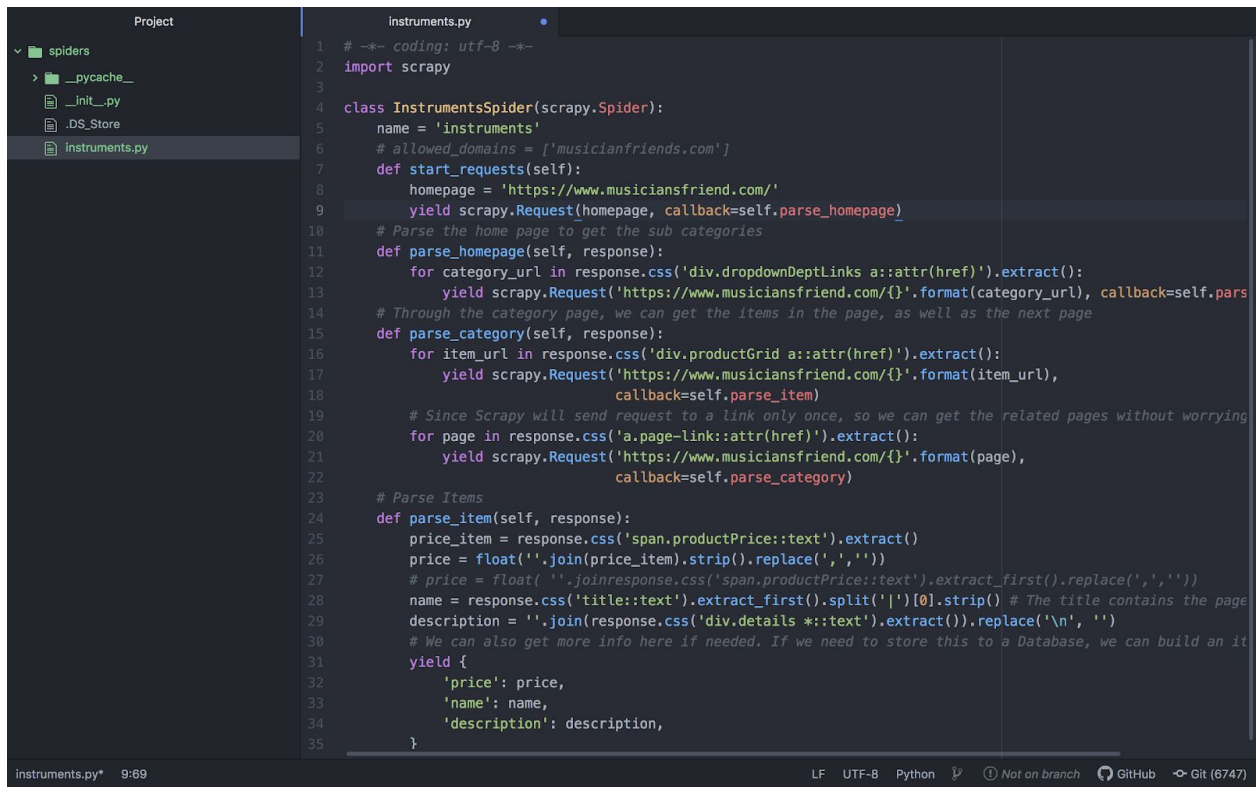
The task we are working on right now is to write a python script using Scrapy that extracts all product related information from stores that primarily sells musical instruments.

After coming up with a list of three large stores that primarily sell musical instruments, such as Musician's Friend, Sweetwater and Reverb, we wrote an initial script program that can extract information from Musician's Friend website. Based on the majority format of all websites, we write our Python program based on the following logic:

1. Set dynamic IP address to solve Anti-Spider for most website settings
2. Go to the website home page (<https://www.musiciansfriend.com/>) and extract according URL for different instrument sub-categories (Guitar, Bass etc.)
3. Iteratively go to each sub-category website and for its listing products, get the detailed URL for every product; Since there are multiple pages for one sub-category, get next-page token and repeat this.
4. For every product page, retrieve product title, product price and product description; Save all data in a CSV file.

For Musician's Friend Website, we finally extract 25505 records of product and its related info in total. Currently, we are working on how to modify this Python program and conform this program to hit all 5 different Musical Instrument stores with slightly different website layouts. In the meantime, we are trying to connect AWS S3 database with our Python script so that it can achieve real-time data storage. Below is the screenshot for part of our Spider program.

Phase 3 Project Report  
Group Mulberry  
Yichang Chen, Chuanjie Xiang, Hanjiao Zhang



```
1  # -*- coding: utf-8 -*-
2  import scrapy
3
4  class InstrumentsSpider(scrapy.Spider):
5      name = 'instruments'
6      # allowed_domains = ['musiciansfriend.com']
7      def start_requests(self):
8          homepage = 'https://www.musiciansfriend.com/'
9          yield scrapy.Request(homepage, callback=self.parse_homepage)
10         # Parse the home page to get the sub categories
11         def parse_homepage(self, response):
12             for category_url in response.css('div.dropdownDeptLinks a::attr(href)').extract():
13                 yield scrapy.Request('https://www.musiciansfriend.com/{}'.format(category_url), callback=self.parse_category)
14             # Through the category page, we can get the items in the page, as well as the next page
15             def parse_category(self, response):
16                 for item_url in response.css('div.productGrid a::attr(href)').extract():
17                     yield scrapy.Request('https://www.musiciansfriend.com/{}'.format(item_url),
18                                         callback=self.parse_item)
19                 # Since Scrapy will send request to a link only once, so we can get the related pages without worrying
20                 for page in response.css('a.page-link::attr(href)').extract():
21                     yield scrapy.Request('https://www.musiciansfriend.com/{}'.format(page),
22                                         callback=self.parse_category)
23             # Parse Items
24             def parse_item(self, response):
25                 price_item = response.css('span.productPrice::text').extract()
26                 price = float(''.join(price_item).strip().replace(',',''))
27                 # price = float(''.join(response.css('span.productPrice::text').extract_first().replace(',','')))
28                 name = response.css('title::text').extract_first().split('|')[0].strip() # The title contains the page
29                 description = ''.join(response.css('div.details *::text').extract()).replace('\n', '')
30                 # We can also get more info here if needed. If we need to store this to a Database, we can build an it
31                 yield {
32                     'price': price,
33                     'name': name,
34                     'description': description,
35                 }
```

## Timeline

- Week 1 and 2: Define problem - Have the initial meeting with Mulberry in person.
- Week 3 and 4: Collect data - Write a spider to automatically extract all qualified product information from five different musical instrument store websites.
- Week 5, 6 and 7: Build initial product category classification model - Multinomial Naive Bayes Classifier
- Week 8, 9 and 10: Improve KPI - Add different features into our initial model and choose the one with a high “coverage ratio”, which is the percentage of warrantable products in a

Phase 3 Project Report  
Group Mulberry  
Yichang Chen, Chuanjie Xiang, Hanjiao Zhang

store that are being successfully returned as a warranty, and in the meanwhile, has few bad predictions.

- Week 11 and 12: Increase the prediction speed - Improve our classification system so that the result could be returned in real-time, ideally under 100 ms, but under 150-200 ms is also acceptable.
- Week 13 and 14: Complete work - Deliver final project and make presentation to our client.

This week, our group will meet Ali Chaudry in Mulberry to talk about data we have already collected and start building our initial model. We aim to build our initial model within one week so that we could have more time to optimize it, or if there is something that goes wrong, it's not too late to correct it.