Title: Mary Jane, the Model: Using Machine Learning to Predict Marijuana Dispensary Performance Author: Scott Butters Date: 2019-04-20 14:15 Modified: 2019-04-21 12:10 Category: blog Tags: python, data science, web scraping, leafly, marijuana, washington Slug: mary-jane-model Summary: In 2012, Washington state passed I-502 and legalized the recreational sale, use, and possession of marijuana. This event has led to an explosion of development in the field that's making waves through our society. Since 2014, approximately 500 state licensed dispensaries have opened throughout the state, with nearly 150 of those here in Seattle. In this project I scour the web for publicly available data that might be predictive of how a cannabis dispensary performs, such as customer reviews, inventory distributions, and local demographics. I then train machine learning models to predict a dispensary's monthly revenue and analyze the resulting models to distill insights about what drives sales in the marijuana market. Status: published

Marijuana is a controversial subject, to be sure. That said, as Washington state and others have legalized recreational pot over the past handful of years the industry has gone from being confined to the shadows to operating on full display in showrooms reminiscent of Apple stores and luxury car dealerships.

Budtender Austin Tucker at Dockside Cannabis in SODO, courtesy of Leafly

For the last two weeks, I've been immersed in a cloud of data surrounding the Washington marijuana industry, attempting to model the relationship between factors like online reviews and local demographics and a dispensary's reported revenue. While it's been a bit hazy at times, it's ultimately led to some enlightening insights. I'd love to share with you some of what I've learned while dabbling with Mary Jane, the model.

## Abstract

In 2012, Washington state passed I-502 and legalized the recreational sale, use, and possession of marijuana. This has led to an explosion of development in the field that's making waves through our society. Since 2014, approximately 500 state licensed dispensaries have opened throughout the state, with nearly 150 of those here in Seattle. In this project, I scoured the web for publicly available data that might be predictive of how a cannabis dispensary performs, such as customer reviews, inventory distributions, and local demographics. I then trained machine learning models to predict a dispensary's monthly revenue and analyze the resulting models to distill insights about what drives sales in the marijuana market. For the full source code from my project, check out my GitHub.

## Tools

**Code**

- Python

- Jupyter notebook

**Data exploration and cleaning**

- Numpy
- Pandas
- Fuzzywuzzy

**Modeling**:

- Sklearn

**Visualization**:

- Matplotlib
- Seaborn

**Web scraping**

- Requests
- Selenium

**Workflow management**

- DataScienceMVP
- Cookiecutter

## Data Sources

- Washington State Liquor and Cannabis Board (WSLCB)
- Leafly
- WA HomeTownLocator
- Walkscore

### Licensing and Sales Data from WSLCB

Lastly, all that data would get us nowhere if we didn't have any target data to train our models on. That's where the WSLCB comes in. The WSLCB maintains data on every dispensary in the state, including monthly reports of revenue (which is what our model is predicting). Their data is scattered across a couple of different outlets, but for this project I used spreadsheets downloadable from this obscure page to get sales data dating back to November 2017. Because the only identifying information in that spreadsheet is the license number of the dispensary, I also downloaded a spreadsheet listing metadata for every entity that has applied for a Marijuana license, which I then joined with the sales data in order to link it up with data scraped from other resources.

### Dispensary profiles from Leafly

Leafly is an information aggregator for cannabis. They maintain a profile for most of the dispensaries in the state. As part of my dataset, I've scraped the following features from the Leafly website for each dispensary for which it was available:

- Average customer rating and number of customer reviews
- Inventory counts (number of products under descriptions like "flower", "edibles", "concentrates", etc.
- Categorical qualities, such as whether or not the store is ADA accessible or has an ATM onsite
- Metadata such as name, address, phone number, etc.

The combination of these features gives us a profile of each dispensary that allow us to draw insights from our model into what makes for a successful dispensary.

### Demographics from WA HomeTownLocator

Of course, having the best inventory, friendliest staff and prettiest pot shop in the state doesn't amount to anything if a dispensary is in the middle of nowhere. This is where demographic data comes in. WA HomeTownLocator maintains a database of demographic statistics for nearly every zip code in the state of Washington. The data is produced by Esri Demographics, and updated 4 times per year using data from the federal census, IRS, USPS, as well as local data sources and more. From this website I scraped data likely to be predictive of a local market such as:

- Population density
- Diversity
- Average income

These data give our model an image of what a dispensary's customer base is like, allowing us to characterize what makes for a good location to establish a dispensary.

### Walkscore

I also used the Walkscore API to query their database for scores on how easily consumers can reach each dispensary by foot or bike.

## Obtaining the Data

The methods for acquiring the data for this project really varied significantly depending upon the source. Here I'll talk briefly about the methods I used for each source and touch on the difficulties and learning experiences I had with each.

## WSLCB License and Sales Data

Other than actually finding this data, this was arguably the easiest piece of data to acquire for this project. If you go down the path of searching for dispensary sales data in the obvious place, which is on the state's Marijuana Dashboard, you'll be disappointed to discover that they basically stopped updating these stats on that page in 2017. No, it turns out that you instead need to go to the WSLCB's Frequently Requested Lists page, which hosts a variety of datasets related to marijuana in the state. Meandering path aside, it's now a matter of simply downloading two spreadsheets—one containing metadata for every marijuana license holder in the state (names, addresses, etc.) and the other containing monthly sales numbers for each license holder dating back to November 2017.

## Scraping Leafly

Getting data I was looking for off of from Leafly's website was by far the most difficult and complex task in the data acquisition portion. To tackle this, I split up the task into two primary functions:

1. Getting a list of all of the dispensaries with a profile on Leafly, along with some basic metadata about each (most importantly, the URL suffix that points to the dispensary's profile page)
2. Going to each individual dispensary profile and extracting specific data of interest available there.

### Getting a list of all of the dispensaries

In a dream world there would be a single page somewhere on Leafly pointing to each of the 500 or so dispensaries in Washington, making it trivial to extract all those links and continue on to get all my data and accomplish great things. This is not that world. Instead, Leafly has this lovely interface of a map view and tiles, dynamically rendered and updated as you move the map or search a new area. Unfortunately, a given map view doesn't actually render all of the dispensaries in that view, nor does the URL interface allow for the map to be easily searched in a programmatic way. Modern javascript allows for some beautiful web design, but it wasn't giving me any handouts here.

Dispensary search page on Leafly.com

That's where a little bit of network traffic sleuthing comes in. It turns out that if you dig into the calls and requests made by the site while the page is loading you can find a searchThisArea API call being made to render the webpage, the results of which include data on every dispensary within a rectangle described by latitude and longitude coordinates. While Leafly doesn't technically offer a public facing API, I was able to exploit this request URL to get what I needed. I wrote a grid search function to systematically enter lat/lon coordinates that would traverse the entire state of Washington one little box at a time.

From here the solution was mostly just a straightforward process of converting request responses to JSON and storing my desired data to a dictionary, I do want to talk briefly about what I thought was a clever response to the API return limit of 200 results per search. Backing up a little bit, in order to avoid being detected as an automated scraping algorithm and having my IP blacklisted from Leafly, my algorithm would wait a random amount of time (anywhere from 0.5-2.5 seconds) between each request in order to obfuscate the fact that it's a data-sucking robot. The downside of this for me is that the smaller my search area in my grid search, the longer it would take me to scrape the whole state.

My solution to that was to implement what approximately amounts to a recursive tree search of the state that tends to roughly minimize the number of API calls required to collect all of the data. I initialized the search by instructing my algorithm to search the entire state of Washington. This invariably returns data on exactly 200 dispensaries, as that's where the API limits the call—problematic when what's desired is *all* of the dispensary. So I set up my routine such that the first thing it does is check whether or not the number of responses equals 200. If so, it simply subdivides the search area into four smaller search areas and searches them by the same routine. This has the effect of automatically determining an appropriately zoomed in search area when moving over dense areas like Seattle while searching from zoomed way out while moving over the more rural parts of the state.

## Getting dispensary specific data

So that was fun, and by the end of the routine I had a data dictionary with entries for a little over 600 dispensaries (my coordinates overlaps into Oregon a bit, and from the looks of it the overachieving stoners down in Portland out-consume Seattle by about 2:1).

From here, I had a new scraping algorithm to write. To get the data off of each individual dispensary page, I used Selenium to fire up an instance of Chrome to be programmatically driven to each page and select and scrape the needed information. Of course, the first step in the process was to have my robobrowser tell Leafly that it was indeed over 21 years of age.

Minor defense on Leafly.com

Simple enough, just had to tell the robot to find the Yes button and click it, and…success! My robot successfully thwarted their robot's defense perimeter and we're off to the races. On each page, I pulled every bit of data I could find that might be relevant, from continuous variables like ratings, number of reviews, and quantity of each product found to be in stock to the categorical things like whether or not they had an ATM on site or are ADA accessible.

Dispensary profile page on Leafly.com

And that was that! With a couple dozen fields of data on each dispensary now stashed away in a JSON file, I could move on to scraping some simpler sources of data. I promise I'll describe those ones more succinctly.

## Scraping demographic data

Getting demographic data for each dispensary area was actually a bit more challenging than I expected. While US census data is publicly available, it's generally not provided at a very granular or intuitive level. After a good long time searching around different aggregator sites, I came across the Washington Hometown Locator website, which offers up data down to each zip code in the state.

Zip code level demographic data from Washington Hometown Locator

From here, I wrote a simple script that would:

1. Extract every zip code from my dataset of dispensary license holders.
2. Generate a URL using each of these zip codes to point to the appropriate page on WHL
3. Download the table data from each of these pages using the Pandas function pd.read_html. It was actually kind of miraculous to me how simple and effective this method was. Thanks to Young Jeong for the tip!

### Getting walk and bike scores from the Walkscore API

It seemed plausible to me that I could leverage Walkscore's scoring of how prime a location is according to how walkable it is, and fortunately for me it they've got a public facing API that's free to access (up to a 5000 requests a day). This one was actually as simple as applying for an API key (took about an hour to get it) and writing a script to generate request call URLs by combining address, city, state, zip code, and the lat/lon coordinates for each location. The only complication is that I didn't yet actually have any single data source where providing me with all of this information. For that small reason I actually don't execute this step until late in the code after I've already done quite a bit of cleaning and merging of the data.

## Cleaning, Combining, and Developing the Data

### Merging data without a common key

### Determining dispensary density

## Explore, Engineer, and Iterate

### Heatmap of correlations

### Regression plots of each feature with target variable

## Building and Refining a Model

### Building an MVP (Minimum Viable Product) with multivariate linear regression

Once I had all my data combined and formatted into a design matrix, it was time to run my first linear regression and get an idea of my baseline model performance. Mind you, the purpose here is not yet to have a particularly *good* model, but simply to have established a simple, functioning pipeline that we can quickly and easily iterate on.

Of course, I couldn't simply allow myself to just throw *all* of my features into a linear regression model and call that good—at this point I was looking at 60 continuous variables, many of them highly covariant, and had just shy of 400 observations in my data. A ratio like that is a certain recipe for an unstable solutionspace that overfits every time, and I wouldn't wish it even on the model of my worst enemy. So I ran a quick

```
target = 'Total Sales'
X = df.corr().sort_values(by=target, ascending=False).drop(target,
axis=1).head(15)
y = df[target]
```

and voila! I had myself a dataframe containing only my 15 features most highly correlated with my target variable. Sure, 15 was an arbitrary choice that just felt a bit better than 60 and I knew there were all kinds of imperfections in the data…but this was something I could regress on with a bit less guilt.

From there I generated an 80/20 train/test split (with a fixed random seed for repeatability, of course), fit a vanilla linear regression model on the X, y values from my training set and then used the model to predict the y values based on only the X from my training set. Unfortunately the precise results from my first run have been lost to the iterative ages, but if memory serves my initial training R^2 came out to about 0.3, with my test R^2 a bit closer to 0.2. I stuck the last few steps in a for-loop and iterated through 50 random seeds to generate a plot much like the following to show me how much it was varying just based on how the dataset was randomly split. Note that the plot below was actually generated after I had already started a bit of feature engineering.

Scoring metrics from initial regression run prior to data processing and feature engineering

At this point I could only loosely argue that my model was performing better than if had simply predicted the mean dispensary revenue every time. Plenty of room to improve!

### Optimizing our model and feature selection with feature scaling and regularization through lasso regression

## So What Insights Can We Glean?

### Descaling and translating our coefficients into terms we can understand

## Future Work

- Collect more data
- More extensive exploratory data analysis
- Time series projections
- Look at revenue *growth* as a target variable
- Experiment with different models
- Suggest optimal locations and product lines