**Problem statement and hypothesis: What factors correlate with a seller's listed price for online housing listings?**

With its diverse metro area, Washington DC has a plethora of housing rental options. During the process of looking for a new apartment, I realized I lacked a sufficient means to analyze the value of an apartment. Hoping to change this, I decided to conduct an analysis of the housing rental market in Maryland, Virginia, and Washington DC.

I have never owned a property and lack an understanding from the seller’s viewpoint. To change this, my project focuses on housing listing data from Craigslist to determine the relative value of different features in the listing. Craigslist does not provide data on final sales prices; thus, using craigslist listings could not provide accurate demand-side metrics on true price of a listing. Yet, the expanse Craigslist data does provide sufficient information on the perceived value from the seller’s standpoint.

More so, I have selected this topic as my final project for the General Assembly Data Science course as it provides vast analysis possibilities. Craigslist data includes text descriptions, preset listing metadata and tags, images, and geolocation data. My goal for the project was to improve my data munging and analysis skillset by conducting a wide range of analyses.

**Data collection and description**

My intent for the final project was to practice the many topics covered in the General Assembly Data Science curriculum. Therefore. I choose to work with a messy user-created dataset to gain practical experience with data collection methodologies and tools. To obtain my dataset I took a two-step process, building a web scraper to acquire online housing listings, and then collecting geographically relevant data from the Google Places API.

**Data Scraping:**

Data scraping is…

I choose to develop a data scraper for Craigslist housing listings. My initial analysis uncovered that the HTML and CSS tags on Craigslist were well formated, lending themselves to work well with a data scraper. The robots.txt files at <http://www.craigslist.org/robots.txt> did not explicitly disallow the analysis of housing listing data. Most importantly, Craiglist was an ideal choice due to its multiple types of provided data:

#### Housing Description:

-Seller's listed price

-Listing Title

-Listing Text

#### Housing Attributes:

{'availability': u'available now',

'bathroom': u' 1',

'bedroom': u'1 ',

'cat': u'cats are OK - purrr',

'data\_available': u'2015-04-15',

'dog': u'dogs are OK - wooof',

'housing\_type': u'apartment',

'laundry': u'w/d in unit',

'parking': u'attached garage',

'smoking': u'no smoking',

'square\_footage': u'763'}

#### Location Data

{'address': u'123 Random Street',

'city': u'Arlington',

'country': u'US',

'latitude': u'11.111111',

'location\_data\_accuracy': u'10',

'longitude': u'-22.222222',

'state': u'VA'}

#### Image data

-number of images

-size of each image (e.g. 600X450.jpg)

-image link

APIs are

According to Google, “The Google Places API Web Service allows you to query for place information on a variety of categories, such as: establishments, prominent points of interest, geographic locations, and more. You can search for places either by proximity or a text string. A Place Search returns a list of places along with summary information about each place”

Using the latitude and longitude provided in each housing listing, I used the Google Places API to collect relevant information on nearby establishments. My hypothesis was that, while not mentioned in most listings, nearby establishments played a role in the sellers listed price. If correct, I could use this information in my personal housing search to avoid expensive, but not personally relevant establishments; this data would also be relevant for sellers who could justify high prices by listing valued local establishments.

My analysis focused on nearby establishments within a one-mile radius, including grocery stores, gyms, movie theatres, train stations, subway stations, airports, barnes and nobles retailers, Deloitte Consulting Offices, and Starbucks

**Description of any pre-processing steps you took**

The nature of the online listings required extensive data cleaning. In my analysis, I created a separate Ipython Notebook to focus on this endeavor.

Price

My first component of the data cleaning was to work with the listing price. Price was a corner stone to my analysis, so I removed all listings without a listed price as they were not relevant. Next, I detected and removed outliers. Most listing feel in the $2000 price range, yet, several were listed well above $10,000 with no justifiable rationale. I removed these listings as I assume these were cases of the user mistyping the price. On the contrary, many listings had a $1 price. Many of these listings had links to external sites; thus, my hypothesis, is that the $1 price was a strategy used by the seller to have their listings appear at the top when a user sorted by price. I also removed these listings to avoid skewing the data.

Cities and States

Craigslist listings often include detailed geolocation data. Most of the listings included both map from Google and Yahoo, which included latitude, longitude, country, state, and city data. The country and city data required the most cleaning. For both, I followed a four step data cleaning process:

* 1. Turn all instances to lower case to reduce differences in capitalization.
     + e.g. Maryland and MARYLAND both became maryland
  2. Replace all encodings with blank string
     + e.g. washington%2edc became washingtondc
  3. Correct misspellings
     + e.g. mayland and marylnd became maryland
  4. Group logical instances
     + e.g. dc and washington became washingtondc

Square Footage

Most listings provided a square footage metric indicating the area of the housing. These were all provided as strings (e.g. 450ft2), so I replaced the ‘ft2’ for each to make them numeric. I removed several listings which did not follow the craigslist template and included square footage values such as loft. To fill in missing values I calculated and imputed the mean value. Given time, my next step would be to improve my imputation method.

**Feature Engineering**

Feature engineering is

Each listing included a date and time of posting. I collected this data with the hypothesis that more reputable rental agencies would post listings during work hours and private owners at off-peak hours. The data and time metrics were provided together. As a simple feature engineering tasks I split these variables to evaluate date and time separately in my model.

#Turn date into weekday

**Impute Missing Values**

Imputation is

Due to the large amount of variables I collected, most listings were missing a portion of data. Modeling libraries in Python, such as scikit-learn, do not accept blank values. I took a simple approach to impute missing values.

1. For categorical values, I created a field specific ‘No\_Field\_Data’ variable (e.g. No\_City\_Data)
2. For numeric values, I filled all missing values a zero

**Data Transformation**

Python modeling libraries often require numeric data. Variables without a natural ordering must be converted to dummy variables with True (1) or False (0) values. For instance, craigslist provided data whether cats were allowed as text. I converted these values to (1) = yes, cat are allowed or (0) no, cats are not allowed. I conducted this process for a variables with binary answers and used pandas.get\_dummies() function to convert all other categorical variables with more than two options, such as city.

#smoking is not a good metric due to mass of NaN values

data['smoking'] = [0 if smoking == 'no smoking' else 1 for smoking in data.smoking]

**What you learned from exploring the data, including visualizations**

In practice, my data cleaning process was conducted in parallel with the project data visualization. Many of the data cleaning choices were initiated from data visualization discoveries. Nonetheless, I choose to split these to topics into separate Ipython Notebooks for ease of data reproducibility.

By running the describe() function on my data set, I learned about the prevalence of missing data. While most of these were corrected in the data cleaning process, the airport proximity data still had complications. According to the Google Places API, none of the listings were within a one-mile radius of an airport. Since every instance had the same value for this variable, it provided no value; I removed it from the analysis.

To learn about the fields in my data set, I created histograms for the distributions of each variable. One prevalent trend was that the majority of the data from the Google Places API was skewed to the right. Due to these distributions, I became less certain that this data would provide value to the analysis.

My criteria to determine if a variable was useful for the analysis was if could provide significant differences in price. If the inclusion or exclusion of the variable provided no change in predicted price, then it would not help the analysis. To visually examine this criteria, I created a function to group each variable by price and then plot a line chart with price on the y-axis and the variable of interest on the x-axis.

**How you chose which features to use in your analysis**

**Details of your modeling process, including how you selected your models and validated them**

**Your challenges and successes**

**Possible extensions or business applications of your project**

**Conclusions and key learnings**

**May 18: First Draft Due**

**At a minimum**, your project repository on GitHub should contain:

* A draft of your project paper (in the format specified [below](https://github.com/justmarkham/DAT5/blob/master/other/project.md#june-3-project-presentation-2))
* Code, with lots of comments
* Visualizations of your data

**Ideally**, you would also include:

* Draft slides for presentation #2
* Data and data dictionary

Your peers and instructors will provide feedback by May 25, according to [these guidelines](https://github.com/justmarkham/DAT5/blob/master/other/peer_review.md).

**Tips for success:**

* The work should stand "on its own", and should not depend upon the reader remembering your first project presentation.
* The better you explain your project, and the easier it is to follow, the more useful feedback you will receive!
* If your reviewers can actually run your code on the provided data, they will be able to give you more useful feedback on your code. (It can be very hard to make useful code suggestions on code that can't be run!)

**June 3: Project Presentation #2**

Your **project paper** should be written with a technical audience in mind. Here are the components you should cover:

Your project repository on GitHub should contain the following:

* **Project paper:** any format (PDF, Markdown, etc.)
* **Presentation slides:** any format (PDF, PowerPoint, Google Slides, etc.)
* **Code:** commented Python scripts, and any other code you used in the project
* **Visualizations:** integrated into your paper and/or slides
* **Data:** data files in "raw" or "processed" format
* **Data dictionary (aka "code book"):** description of each variable, including units

If it's not possible or practical to include your entire dataset, you should link to your data source and provide a sample of the data. (GitHub has a [size limit](https://help.github.com/articles/what-is-my-disk-quota/) of 100 MB per file and 1 GB per repository.) If your data is private, you can either include an "anonymized" version of your data or create a private GitHub repository.