OH, THE PLACES A DATA SCIENTIST WILL GO!

Michelle Mak

IST 652 Final Project

Overview

For my final project, I will investigate the job market for Data Scientists in anticipation of my impending graduation. Using data scrubbed from Indeed.com and Zillow Research, I will pinpoint optimal locations for potential jobs as well as define the most valuable skills to learn or highlight on my résumé.

This investigation will attempt to answer the following questions:

- Who is hiring data scientists?
- Where are they hiring data scientists?
- What are the most optimal cities for relocation?
- What skills are companies looking for in a data scientist?

Data

A data set found on Kaggle provided 7,000 data science positions scrubbed from Indeed.com on August 3rd, 2018. That data included company name, position, location, job description, and number of reviews of the company.

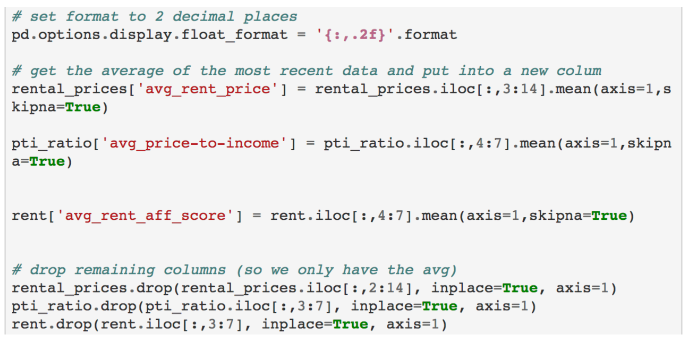
Supplemental data was found on Zillow Research, which provided time series data of the average rental price of a one-bedroom apartment in cities across the United States. In addition, another dataset provided a rent affordability score and price-to-income ratio created by Zillow.

Data Cleaning and Preprocessing

Four datasets in total are read into the notebook:

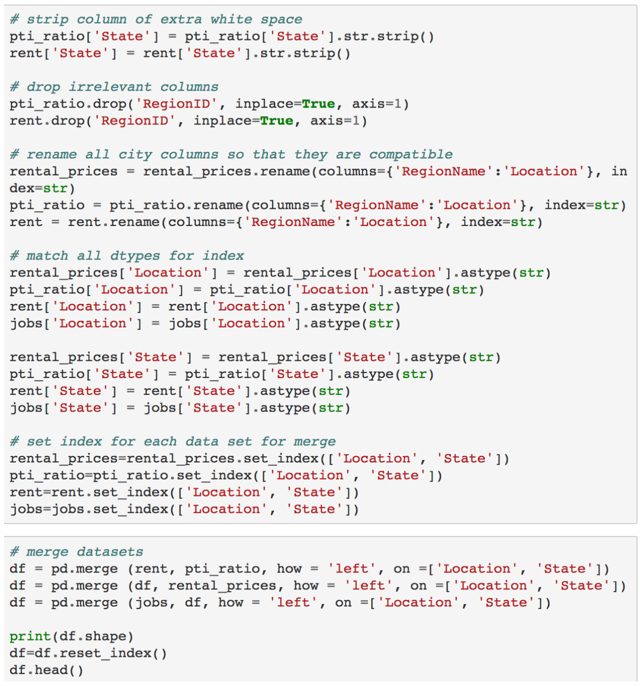
- Zillow: Median rental prices
- Zillow: Price-to-income ratio
- Zillow: Rent affordability score
- Indeed: Data Scientist job postings

Since the Zillow data is all in timer series format, only data from 2018 will be used. All values from 2018 columns are averaged in a new column. These new columns will be the only source of information used from these datasets.



```
# set format to 2 decimal places
pd.options.display.float_format = '{:,.2f}'.format

# get the average of the most recent data and put into a new colum
rental_prices['avg_rent_price'] = rental_prices.iloc[:,3:14].mean(axis=1,s
kipna=True)

pti_ratio['avg_price-to-income'] = pti_ratio.iloc[:,4:7].mean(axis=1,skipn
a=True)


rent['avg_rent_aff_score'] = rent.iloc[:,4:7].mean(axis=1,skipna=True)


# drop remaining columns (so we only have the avg)
rental_prices.drop(rental_prices.iloc[:,2:14], inplace=True, axis=1)
pti_ratio.drop(pti_ratio.iloc[:,3:7], inplace=True, axis=1)
rent.drop(rent.iloc[:,3:7], inplace=True, axis=1)
```

Next, the state columns in each data frame are stripped of any leading or trailing whitespace, which will prevent the datasets from merging correctly. Irrelevant columns such as 'Region ID' are dropped, and 'RegionName' is renamed to 'Location' so that it is easily mergeable with Indeed data. The index is set to 'Location' and 'State' for all data frames and then merged together to form the working data frame.

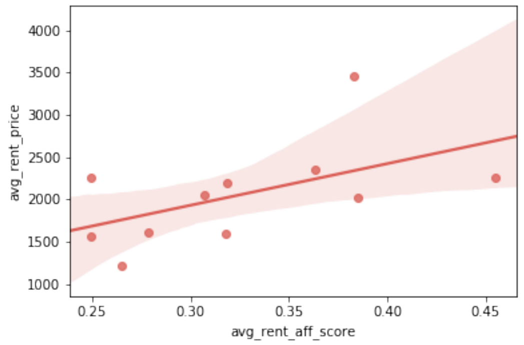


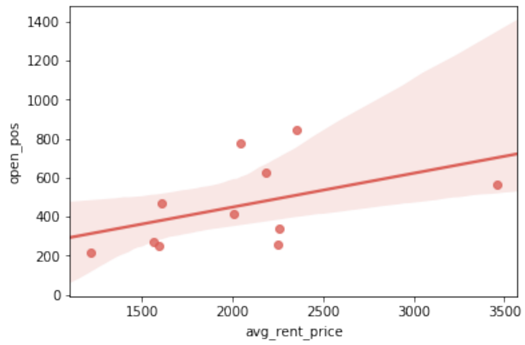
```
# strip column of extra white space
pti_ratio['State'] = pti_ratio['State'].str.strip()
rent['State'] = rent['State'].str.strip()

# drop irrelevant columns
pti_ratio.drop('RegionID', inplace=True, axis=1)
rent.drop('RegionID', inplace=True, axis=1)

# rename all city columns so that they are compatible
rental_prices = rental_prices.rename(columns={'RegionName':'Location'}, in
dex=str)
pti_ratio = pti_ratio.rename(columns={'RegionName':'Location'}, index=str)
rent = rent.rename(columns={'RegionName':'Location'}, index=str)

# match all dtypes for index
rental_prices['Location'] = rental_prices['Location'].astype(str)
pti_ratio['Location'] = pti_ratio['Location'].astype(str)
rent['Location'] = rent['Location'].astype(str)
jobs['Location'] = jobs['Location'].astype(str)

rental_prices['State'] = rental_prices['State'].astype(str)
pti_ratio['State'] = pti_ratio['State'].astype(str)
rent['State'] = rent['State'].astype(str)
jobs['State'] = jobs['State'].astype(str)

# set index for each data set for merge
rental_prices=rental_prices.set_index(['Location', 'State'])
pti_ratio=pti_ratio.set_index(['Location', 'State'])
rent=rent.set_index(['Location', 'State'])
jobs=jobs.set_index(['Location', 'State'])
```

```
# merge datasets
df = pd.merge (rent, pti_ratio, how = 'left', on =['Location', 'State'])
df = pd.merge (df, rental_prices, how = 'left', on =['Location', 'State'])
df = pd.merge (jobs, df, how = 'left', on =['Location', 'State'])

print(df.shape)
df=df.reset_index()
df.head()
```

Methods

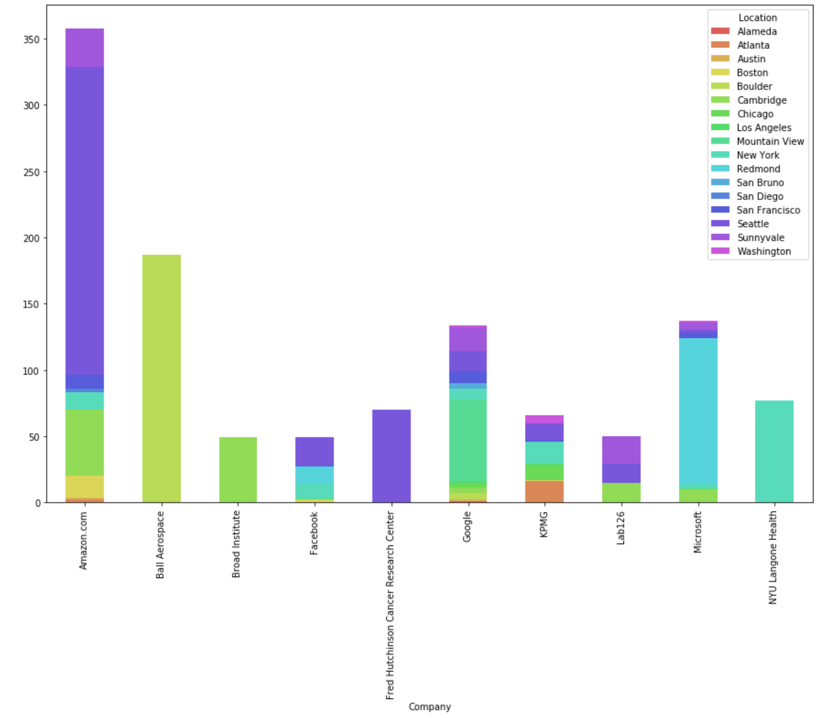
Some initial data exploration reveals that average affordability score and average rental price both have slight positive correlations; as does average rent price and number of open positions per city.





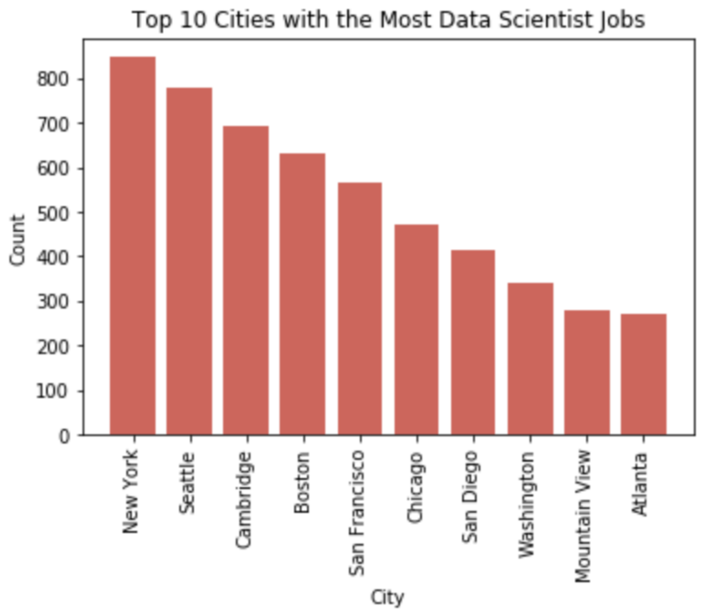
These results are not surprising. However, based on the second line plot, it looks like there are some cities with a healthy amount of opportunities at average rent prices. The goal is to pinpoint those as possible options.

To answer the first question, “Where are they hiring data scientists?”, a count of all the different companies in the data frame is achieved using the counter package. Amazon, Ball Aerospace, Microsoft, Google, and NYU Langone Health are the top 5 companies hiring data scientists. Next, by aggregating the Company and Location data together, a stacked bar plot displaying the cities in which each of these companies are hiring is achieved.

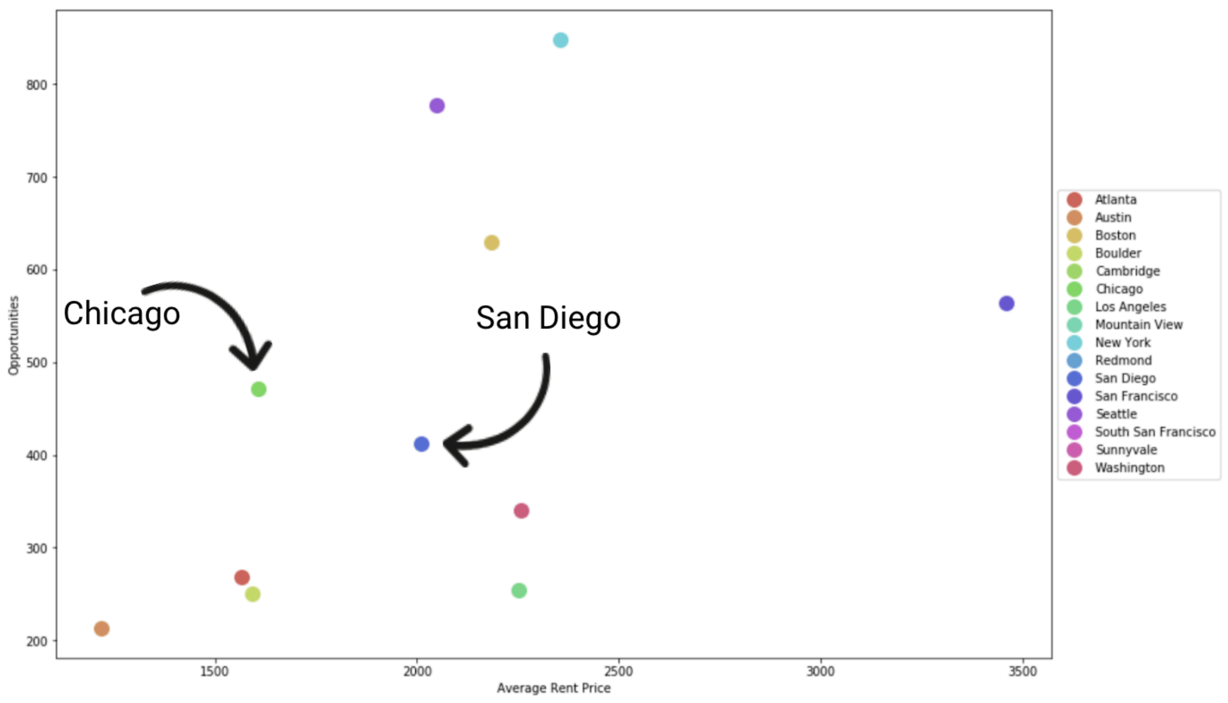


Interestingly, in exploring the next question (Where are they hiring data scientists?) the cities that were relevant in the initial bar plot are also the same cities that are hiring the most data scientists.

However, while Amazon was hiring mostly in Seattle, New York actually has the most data scientist jobs.



To add another dimension to this investigation, the variable of average rental price in each city is introduced. A pivot view of the data is created using the count of the job opportunities per city and grouped by city to create the following scatter plot.



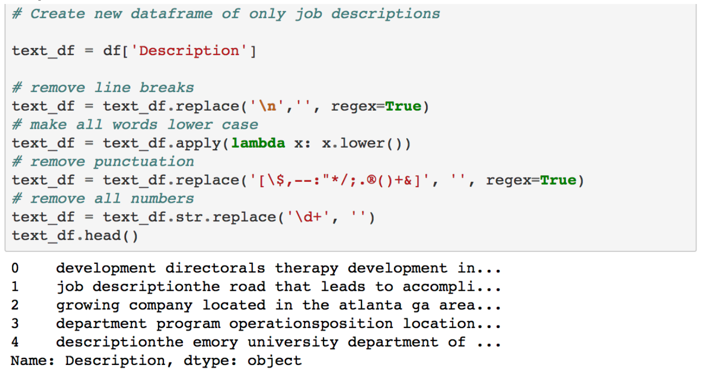
Since my goal is to find somewhere that is affordable but will have enough opportunities explore, my top two choices ended up being San Diego and Chicago, since their average rental prices are on the lower half of the spectrum.

Text Mining

In order to investigate the types of skills I should highlight in my résumé, some text mining methods were used to explore the job descriptiongs.

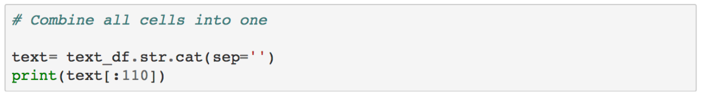
Data Cleaning and Preprocessing

First, the 'Description' column of the working data frame is isolated into a new data frame. Next, all line breaks, punctuation, and numbers are removed from the text. All text is also converted to lower case. The cells of each job description are then combined into a single block of text to create the corpus.



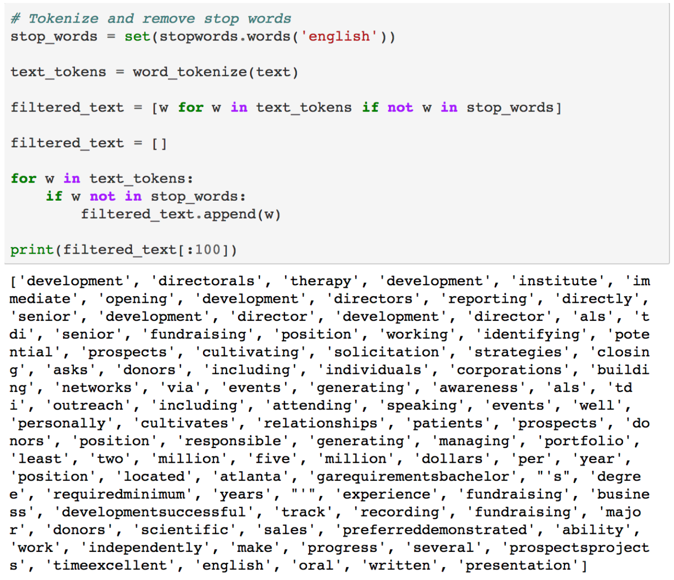
```
# Create new dataframe of only job descriptions

text_df = df['Description']

# remove line breaks
text_df = text_df.replace('\n','', regex=True)
# make all words lower case
text_df = text_df.apply(lambda x: x.lower())
# remove punctuation
text_df = text_df.replace('[\$,--:"*/;.®()+&]', '', regex=True)
# remove all numbers
text_df = text_df.str.replace('\d+', '')
text_df.head()
```

```
0    development directorals therapy development in...
1    job descriptionthe road that leads to accompli...
2    growing company located in the atlanta ga area...
3    department program operationsposition location...
4    descriptionthe emory university department of ...
Name: Description, dtype: object
```



```
# Combine all cells into one

text= text_df.str.cat(sep='')
print(text[:110])
```

Each word in the corpus is then tokenized and a function is created to remove all stop words from the default dictionary. This will accomplish the necessary preprocessing for text mining.

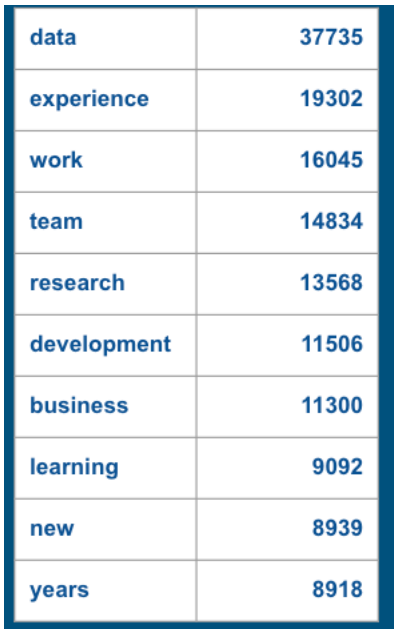


```
# Tokenize and remove stop words
stop_words = set(stopwords.words('english'))

text_tokens = word_tokenize(text)

filtered_text = [w for w in text_tokens if not w in stop_words]

filtered_text = []

for w in text_tokens:
    if w not in stop_words:
        filtered_text.append(w)

print(filtered_text[:100])
```

```
['development', 'directorals', 'therapy', 'development', 'institute', 'im
mediate', 'opening', 'development', 'directors', 'reporting', 'directly',
'senior', 'development', 'director', 'development', 'director', 'als', 't
di', 'senior', 'fundraising', 'position', 'working', 'identifying', 'pote
ntial', 'prospects', 'cultivating', 'solicitation', 'strategies', 'closin
g', 'asks', 'donors', 'including', 'individuals', 'corporations', 'buildi
ng', 'networks', 'via', 'events', 'generating', 'awareness', 'als', 'td
i', 'outreach', 'including', 'attending', 'speaking', 'events', 'well',
'personally', 'cultivates', 'relationships', 'patients', 'prospects', 'do
nors', 'position', 'responsible', 'generating', 'managing', 'portfolio',
'least', 'two', 'million', 'five', 'million', 'dollars', 'per', 'year',
'position', 'located', 'atlanta', 'garequirementsbachelor', "'s", 'degre
e', 'requiredminimum', 'years', "'", 'experience', 'fundraising', 'busine
ss', 'developmentsuccessful', 'track', 'recording', 'fundraising', 'majo
r', 'donors', 'scientific', 'sales', 'preferreddemonstrated', 'ability',
'work', 'independently', 'make', 'progress', 'several', 'prospectsproject
s', 'timeexcellent', 'english', 'oral', 'written', 'presentation']
```

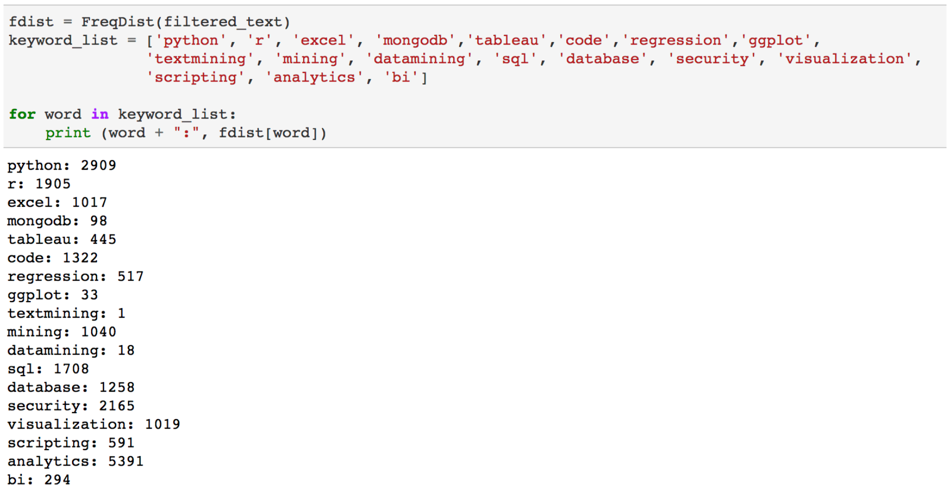
Methods

A frequency distribution of the tokenized words is created to see the top 50 most common words in all the job descriptions. This list was surprisingly inconclusive as the top 50 words were pretty general and did not include any specific skill set that has been taught during my grad program.



data	37735
experience	19302
work	16045
team	14834
research	13568
development	11506
business	11300
learning	9092
new	8939
years	8918

Therefore, a keyword list is created in order to identify specific skill sets hypothesized to be important for data scientist positions. The keyword list is applied to the frequency distribution to see which ones are the most relevant and important.



```
fdist = FreqDist(filtered_text)
keyword_list = ['python', 'r', 'excel', 'mongodb','tableau','code','regression','ggplot',
               'textmining', 'mining', 'datamining', 'sql', 'database', 'security', 'visualization',
               'scripting', 'analytics', 'bi']

for word in keyword_list:
    print (word + ":", fdist[word])
```

```
python: 2909
r: 1905
excel: 1017
mongodb: 98
tableau: 445
code: 1322
regression: 517
ggplot: 33
textmining: 1
mining: 1040
datamining: 18
sql: 1708
database: 1258
security: 2165
visualization: 1019
scripting: 591
analytics: 5391
bi: 294
```

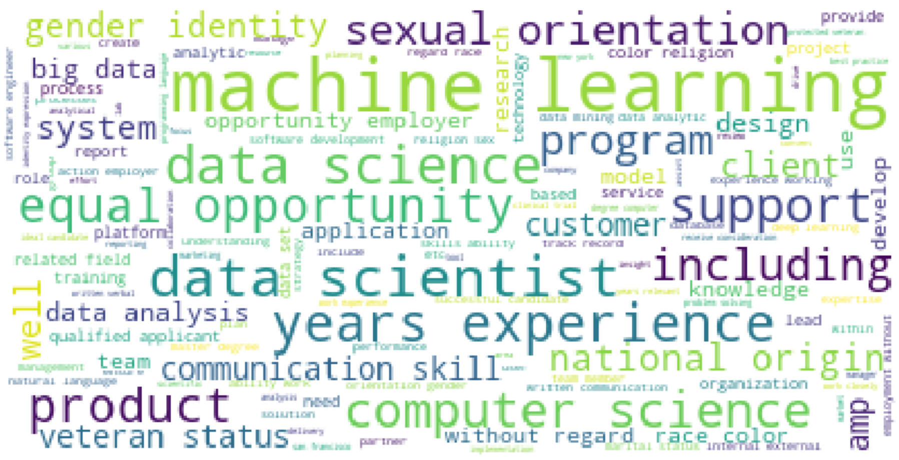
This list is then divided by the total number of words in the corpus to find the normalized percentage of how often the word appears. The word that appears the most in the keyword list is analytics at only 0.23% of the total words. The next most frequent word in the keyword list is python at 0.12%. This is the most surprising finding in this investigation as it was assumed that these words would be far more relevant to the job descriptions.

Finally, a bigram is also investigated to see if there are any specific phrases that could provide more insight.

```
(('machine', 'learning'), 0.002764617686022271)
(('data', 'science'), 0.001353853948084579)
(('computer', 'science'), 0.0011668646385672083)
(('years', 'experience'), 0.0010889880840313697)
(('equal', 'opportunity'), 0.0010380687983733213)
(('sexual', 'orientation'), 0.0009019987745139989)
(('national', 'origin'), 0.0009002872018868377)
(('gender', 'identity'), 0.0008536468477966926)
(('data', 'scientists'), 0.0008125691047448216)
(('veteran', 'status'), 0.0007513803833238056)
(('communication', 'skills'), 0.0007239952212892225)
(('data', 'analysis'), 0.0007235673281324347)
(('’', 'degree'), 0.0007231394349756443)
(('big', 'data'), 0.0007201441828781121)
(('race', 'color'), 0.0006889079824324185)
(('data', 'scientist'), 0.0006752154014151283)
(('opportunity', 'employer'), 0.0006709364698472225)
(('without', 'regard'), 0.0005947714879385477)
(('qualified', 'applicants'), 0.0005866415179595315)
(("'s", 'degree'), 0.0004659756477446073)
(('related', 'field'), 0.0004646919682742898)
(('color', 'religion'), 0.00045442253251132205)
(('software', 'development'), 0.0004432973104347737)
(('data', 'sets'), 0.00044286941727798336)
```

Interestingly enough, ‘machine learning’ is the top phrase. Followed by ‘data science’ and ‘computer science.’ It is interesting that computer science appears so often, but seeing as there have not been a lot of data science up until recently, computer science seems to be the substitute program that feeds into data science roles.

A word cloud is created for a better visual of the word distribution:



Conclusion

Based on the findings from this investigation, San Diego and Chicago seem to be the optimal locations to look for jobs due to the fact that there are abundant opportunities and average rental prices. In terms of skillsets to highlight, machine learning and data science are great key words. Python seems to be the most important skill to have based on the frequency distribution of job description words.

Further Recommendations

Deeper investigation into this topic can include weather data of each city as well as salary data. These are two important factors that should definitely be taken into consideration when looking for job opportunities. Furthermore, looking for positions from multiple forums might change the data as different companies might post to different sites, thereby diversifying that opportunities. If possible, company ratings and reviews would also be an interesting new layer to add onto this exploration.