

# Vibe of Boston

Vedant Rautela & Marc Bacvanski

---

## Description of Need

We were inspired by the question, “Where should I stay when I visit Boston?” When we visit a new city, we do not know what to expect, but we are looking for a place to stay that matches our “vibe” – our individual preferences in terms of environment, attitude, and emotional state. No single “Guide to Visiting Boston” will suffice for all of us, since our preferences are very individual and unique.

19 million people visit Boston every year, with 1.3 million of those people being from overseas. While Boston is smaller than many similarly visited cities, Boston is divided into 23 official neighborhoods, each of which offers a different living experience. Boston’s set of neighborhoods mean that the city has a very diverse set of places to stay. Each neighborhood has its own personality, environment, and overall “vibe” than other neighborhoods, and people may prefer different qualities of the place they live in. We want to answer the question of where a visitor would enjoy staying, to maximize their enjoyment.

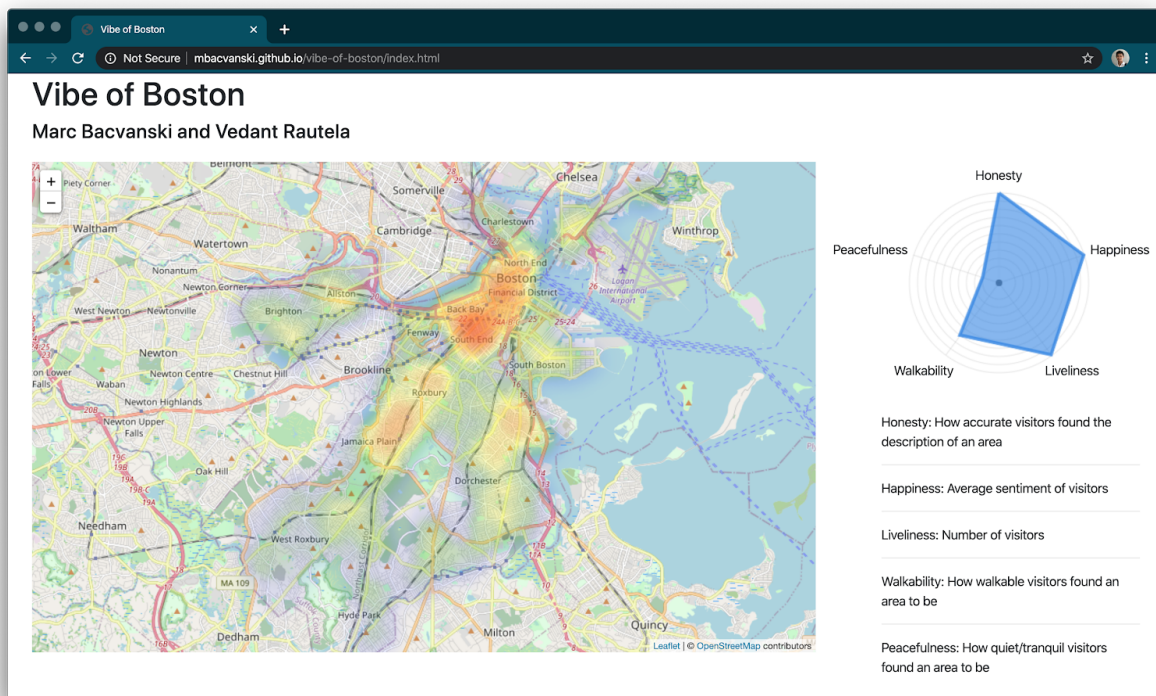
The purpose of our product is to help prospective visitors to Boston decide where to live, rent, or explore. Users of our tool will be able to uncover data-driven insights about neighborhoods based on the experiences of both those who live there, as well as those of visitors. Users will be able to use that information to find the places in Boston that most match their vibe.

## Functionality

“Vibe” is a very vague word, and in the context of analyzing the “vibe of Boston”, we wanted to offer the user the ability to quantify and personalize this emotional response.

The tool that we developed to help users to answer this question is an application which calculates and visualizes on a heatmap, the areas of Boston which the user would likely enjoy most.

For each listing, we calculate a “vibe vector”, composed of our calculations of a number of vibe factors. The five vibe factors that we’ve chosen to include on the heatmap are *Honesty*, *Happiness*, *Liveliness*, *Walkability*, and *Peacefulness*. By no means are these a comprehensive list of all the features of Boston, but we hope that these factors will be relatively common ones that people look for in determining where to stay.



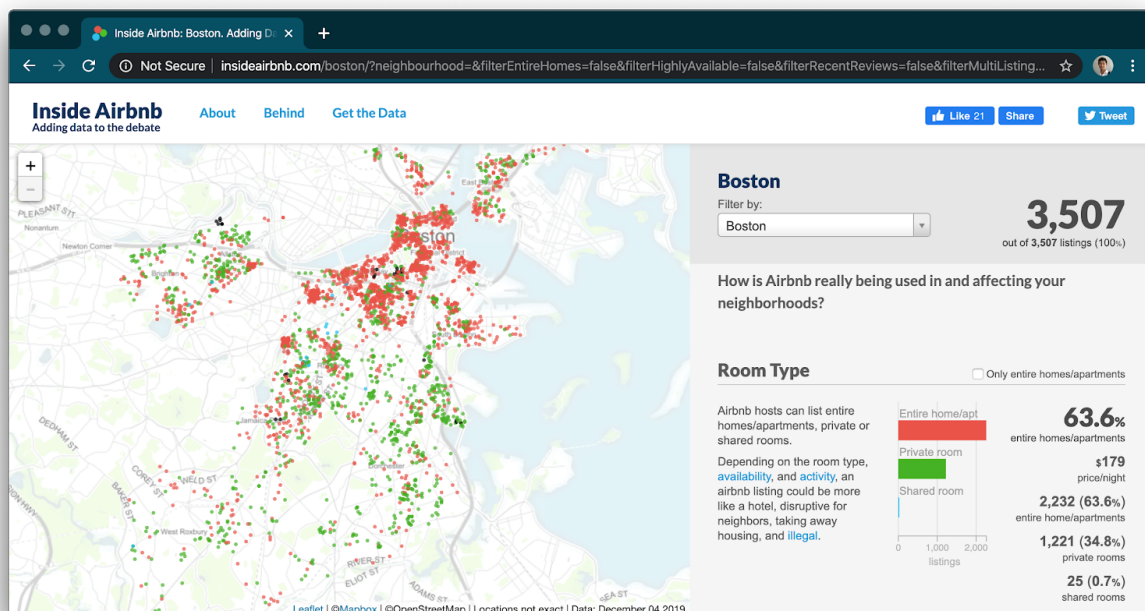
Live demo: <https://mbacvanski.github.io/vibe-of-boston/>

Here, the user has inputted their preferences, giving high weights to *Honesty*, *Happiness*, and *Liveliness*. To this user, *Walkability* matters slightly less, and *Peacefulness* of the neighborhood does not matter. Given this individualized weighting of factors, our application has generated a custom heatmap of the areas of Boston that most match this vibe. The user can move the map around and zoom in, observing how Back Bay, South End, and the Jamaica Plain are highlighted in red, and may be the best places for them to stay. Other areas that the user may find interesting have less intensity, such as North End, South End, and Back Bay.

# Datasets and Analysis

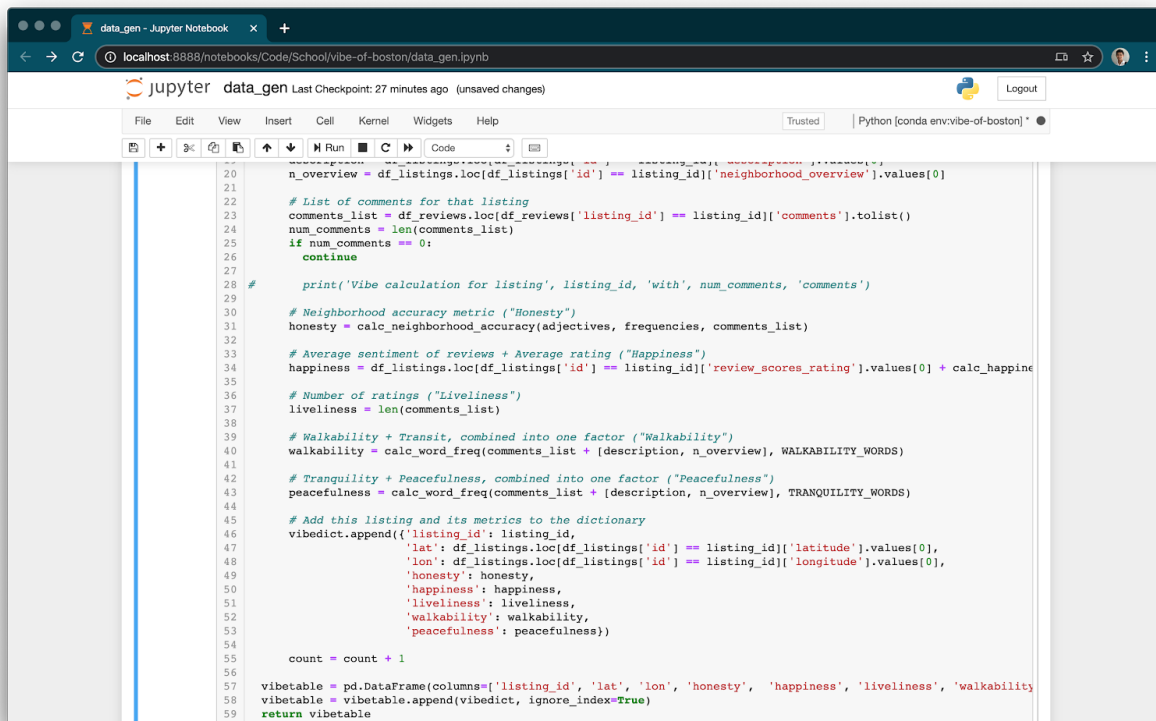
## Airbnb Dataset

We performed our analysis on the [Airbnb dataset](#) of listings, reviews, and bookings in the Boston area. We used the publicly available data from 2015 to 2020, sampled yearly. This yielded us 806,857 unique data points, of which the majority were reviews. We chose the Airbnb dataset since it includes data from both residents of Boston (who write listings) and the visitors to Boston (who write reviews). We hope that the combination of this data will yield insight into how it is to live in Boston, from a diverse and inclusive perspective that users can trust.



## Analysis Approaches

We performed preliminary data analysis in Python. We manipulated data using Pandas dataframes, which offered a performant interface to manipulate the datasets. For preliminary natural language processing, we used the spaCy library's tokenizer and entity recognition due to its high performance and intelligent recognition of common entities. Our Jupyter notebook details our approach to cleaning, analyzing, and preparing the data for visualization.



```
20 n_overview = df_listings.loc[df_listings['id'] == listing_id]['neighborhood_overview'].values[0]
21
22 # List of comments for that listing
23 comments_list = df_reviews.loc[df_reviews['listing_id'] == listing_id]['comments'].tolist()
24 num_comments = len(comments_list)
25 if num_comments == 0:
26     continue
27
28 # print('Vibe calculation for listing', listing_id, 'with', num_comments, 'comments')
29
30 # Neighborhood accuracy metric ("Honesty")
31 honesty = calc_neighborhood_accuracy(adjectives, frequencies, comments_list)
32
33 # Average sentiment of reviews + Average rating ("Happiness")
34 happiness = df_listings.loc[df_listings['id'] == listing_id]['review_scores_rating'].values[0] + calc_happine
35
36 # Number of ratings ("Liveliness")
37 liveliness = len(comments_list)
38
39 # Walkability + Transit, combined into one factor ("Walkability")
40 walkability = calc_word_freq(comments_list + [description, n_overview], WALKABILITY_WORDS)
41
42 # Tranquility + Peacefulness, combined into one factor ("Peacefulness")
43 peacefulness = calc_word_freq(comments_list + [description, n_overview], TRANQUILITY_WORDS)
44
45 # Add this listing and its metrics to the dictionary
46 vibedict.append({'listing_id': listing_id,
47                 'lat': df_listings.loc[df_listings['id'] == listing_id]['latitude'].values[0],
48                 'lon': df_listings.loc[df_listings['id'] == listing_id]['longitude'].values[0],
49                 'honesty': honesty,
50                 'happiness': happiness,
51                 'liveliness': liveliness,
52                 'walkability': walkability,
53                 'peacefulness': peacefulness})
54
55 count = count + 1
56
57 vibetable = pd.DataFrame(columns=['listing_id', 'lat', 'lon', 'honesty', 'happiness', 'liveliness', 'walkability',
58                                'peacefulness'])
59 vibetable = vibetable.append(vibedict, ignore_index=True)
60 return vibetable
```

## Vibe Factors

Natural language processing from the spaCy library was used to do in-depth analysis of the listings and review data to generate a number of metrics and ratings. For each listing, we calculated a vibe vector by computing five different vibe factors. This data is exported as a large JSON file, which is then loaded by our user-facing visualization tool.

### *Honesty*

Honesty is calculated by extracting the most common adjectives from the listings, and comparing them with the most common adjectives from the reviews. This is a metric of how close each listing is to the average vibe of the neighborhood it is in.

```

1 # Load spacy
2 import spacy
3 nlp = spacy.load('en_core_web_sm')
4
5 # Load Vader sentiment analysis
6 from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
7 analyser = SentimentIntensityAnalyzer()
8
9 # Calculates the neighborhood accuracy statistic for a listing
10 # Neighborhood accuracy is the dot product between the neighborhood description
11 # and the reviews left for the place. This metric is greatest when the experience
12 # of the visitors most matched the vibe that was portrayed by the listing.
13
14 # Neighborhood accuracy is calculated as the sum of the number of adjectives that appear
15 # in both the neighborhood description and in the reviews for a listing,
16 # weighted by the frequency by which those adjectives occur in all
17 # neighborhood descriptions within that neighborhood.
18
19 # calc_neighborhood_honesty(comments_list) takes in a list of strings as the comments
20 # for that listing. It returns a scalar neighborhood accuracy score.
21 def calc_neighborhood_honesty(neighborhood_adjectives, neighborhood_frequencies, comments_list):
22     neighborhood_honesty = 0
23
24     # Comments, excluding punctuation and special characters
25     listing_comments = re.sub('[^\w\s]+', '', ' '.join(comments_list)).split()
26
27     # Count up the frequencies of these words
28     comment_freq = Counter(listing_comments)
29     comment_words = list(comment_freq.keys())
30
31     # Dot product time doo doo doo doo doo
32     for i in range(len(neighborhood_adjectives)):
33         if (neighborhood_adjectives[i] in comment_words):
34             neighborhood_honesty += neighborhood_frequencies[i] * comment_freq[neighborhood_adjectives[i]]
35
36     neighborhood_honesty /= len(comments_list)
37
38     return neighborhood_honesty

```

## Liveliness

Liveliness is the total number of reviews left by visitors to a place. This is a metric of how many people visit and are living in an area.

```

16 # for each listing within this neighborhood
17 for listing_id in listings:
18
19     description = df_listings.loc[df_listings['id'] == listing_id]['description'].values[0]
20     n_overview = df_listings.loc[df_listings['id'] == listing_id]['neighborhood_overview'].values[0]
21
22     # List of comments for that listing
23     comments_list = df_reviews.loc[df_reviews['listing_id'] == listing_id]['comments'].tolist()
24     num_comments = len(comments_list)
25     if num_comments == 0:
26         continue
27
28     # Number of ratings ("Liveliness")
29     liveliness = len(comments_list)

```

## Happiness

Happiness is calculated as the sum of the average rating of a listing and the average sentiment of reviews left by visitors. This is a metric of how lively and desirable an area is.



```

40 # Calculates the mean sentiment of all reviews for a listing.
41 def calc_happiness(comments_list):
42     all_comments = ' '.join(comments_list)
43
44     happiness = analyser.polarity_scores(all_comments)['compound']
45     return happiness
46
47
48 # Average sentiment of reviews + Average rating ("Happiness")
49 happiness = df_listings.loc[df_listings['id'] == listing_id]['review_scores_rating'].values[0]
50 + calc_happiness(comments_list)

```

## Walkability & Tranquility

Both of these factors are calculated with the occurrence of those related words within each of the reviews and the listing description itself. This count is normalized by dividing by the total number of words encountered in the description, neighborhood overview, and reviews to find the average walkability score of a given listing.

```

47 WALKABILITY_WORDS = ["transit", "train", "subway", "T", "bus", "ride", "walk", "walking", "walkable"]
48 TRANQUILITY_WORDS = ["quiet", "peace", "peaceful", "pleasant", "nice"]
49
50 # Calculates the frequency of words within word list appearing in a comments list
51 def calc_word_freq(comments_list, word_list):
52     total_count = 0
53     len_total = 0
54
55     # Comments, excluding punctuation and special characters
56     listing_comments = re.sub('[^\w\s]+', '', ' '.join(comments_list)).split()
57
58     comment_freq = Counter(listing_comments)
59     comment_words = list(comment_freq.keys())
60
61     for word in comment_words:
62         len_total += comment_freq[word]
63         if word in word_list:
64             total_count += comment_freq[word]
65
66     # Normalized by total number of words
67     return total_count / len_total

```

## Vibe Factor Aggregation

```

30 # Neighborhood accuracy metric ("Honesty")
31 honesty = calc_neighborhood_honesty(adjectives, frequencies, comments_list)
32
33 # Average sentiment of reviews + Average rating ("Happiness")
34 happiness = df_listings.loc[df_listings['id'] == listing_id]['review_scores_rating'].values[0] + calc_happiness(comments_list)
35
36 # Number of ratings ("Liveliness")
37 liveliness = len(comments_list)
38
39 # Walkability + Transit, combined into one factor ("Walkability")
40 walkability = calc_word_freq(comments_list + [description, n_overview], WALKABILITY_WORDS)
41
42 # Tranquility + Peacefulness, combined into one factor ("Peacefulness")
43 peacefulness = calc_word_freq(comments_list + [description, n_overview], TRANQUILITY_WORDS)
44
45 # Add this listing and its metrics to the dictionary
46 vibedict.append({'listing_id': listing_id,
47                 'lat': df_listings.loc[df_listings['id'] == listing_id]['latitude'].values[0],
48                 'lon': df_listings.loc[df_listings['id'] == listing_id]['longitude'].values[0],
49                 'honesty': honesty,
50                 'happiness': happiness,
51                 'liveliness': liveliness,
52                 'walkability': walkability,
53                 'peacefulness': peacefulness})
54
55 count = count + 1

```

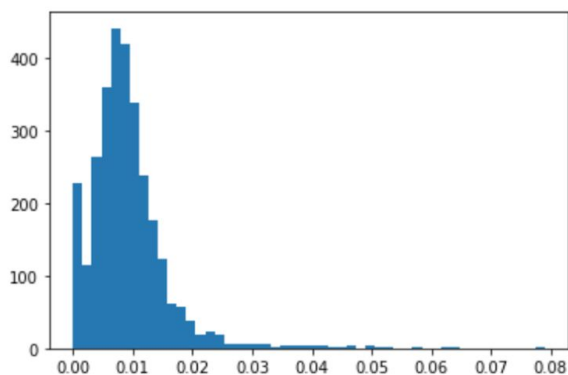
After calculating each of the vibe factors, we add the listing as an entry to a dictionary, which we add all at once to a Pandas dataframe for maximal efficiency.

## Normalization of Vibe Factors

Each of the vibe factors' data points have different ranges. In order to allow the vibe alignment chart to give equal weight to each of the factors, our goal is to scale each of the factors' data points to have values between 0 and 1.

We first examined the factors individually, to visualize their distributions. For example, here is a histogram for the distribution of walkability, before scaling.

```
import matplotlib.pyplot as plt
plt.hist(vibetable['walkability'], bins=50)
plt.show()
```

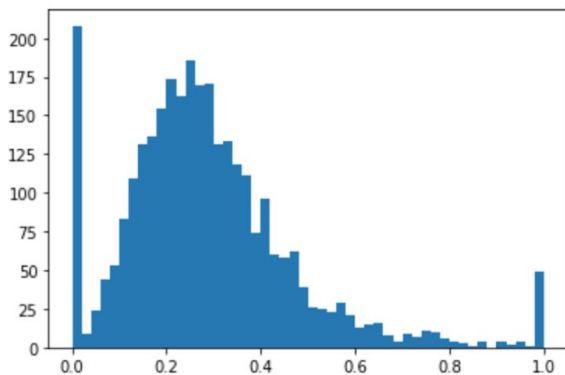


For example, for *Honesty*, we first subtract the 0.01'th quantile from all data points, clipping at 0. We then divide these data points by the 0.99'th quantile, clipping at 1. These quantiles took some hand tuning, but we tried to ensure that the resulting distribution looked very similar to what the data was before scaling, i.e. we were not clipping too many points above or below.

```
1 import numpy as np
2
3 scaled_table = vibetable.copy()
4
5 scaled_table['honesty'] = (vibetable['honesty'] - np.quantile(vibetable['honesty'], 0.01)).clip(lower=0)
6 scaled_table['honesty'] = (scaled_table['honesty'] / np.quantile(scaled_table['honesty'], 0.99)).clip(upper=1)
7
8 scaled_table['happiness'] = (vibetable['happiness'] - np.quantile(vibetable['happiness'], 0.01)).clip(lower=0)
9 scaled_table['happiness'] = (scaled_table['happiness'] / np.quantile(scaled_table['happiness'], 0.99)).clip(upper=1)
10
11 scaled_table['liveliness'] = (vibetable['liveliness'] - np.quantile(vibetable['liveliness'], 0.0000001)).clip(lower=0)
12 scaled_table['liveliness'] = (scaled_table['liveliness'] / np.quantile(scaled_table['liveliness'], 0.99)).clip(upper=1)
13
14 scaled_table['walkability'] = (vibetable['walkability'] - np.quantile(vibetable['walkability'], 0.01)).clip(lower=0)
15 scaled_table['walkability'] = (scaled_table['walkability'] / np.quantile(scaled_table['walkability'], 0.985)).clip(upper=1)
16
17 scaled_table['peacefulness'] = (vibetable['peacefulness'] - np.quantile(vibetable['peacefulness'], 0.01)).clip(lower=0)
18 scaled_table['peacefulness'] = (scaled_table['peacefulness'] / np.quantile(scaled_table['peacefulness'], 0.99)).clip(upper=1)
19
```

Now, here is the final histogram for the distribution of walkability, after scaling. We observe how all values fall nicely between 0 and 1.

```
import matplotlib.pyplot as plt
plt.hist(scaled_table['walkability'], bins=50)
plt.show()
```



## Visualization

To plot the heatmap, we used the [Leaflet](#) library atop a [OpenStreetMap](#) layer.

```
function processVibeTable(data, weights, map) {
  let biggestVibe = 0;
  let formattedData = [];
  Object.keys(data).forEach(key => {
    const point = data[key];
    const vibe = calcVibe(point, weights);
    const arr = [point['lat'], point['lon'], vibe];
    formattedData.push(arr);
    if (vibe > biggestVibe) {
      biggestVibe = vibe;
    }
  });

  map.eachLayer(function (layer) {
    map.removeLayer(layer);
  });
  L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors',
  }).addTo(map);
  const heat = L.heatLayer(formattedData, {
    max: biggestVibe,
    gradient: {0.1: 'blue', 0.2: 'green', 0.3: 'yellow', 0.4: 'orange', 1: 'red'}
  }).addTo(map);
}
```

The interactive vibe factor alignment chart was made using [Spidergraph](#). Changes to the weights in the vibe alignment chart cause the vibe factors to be recalculated.



```

$('#spidergraphcontainer').spidergraph({
  'fields': ['Honesty', 'Happiness', 'Liveliness', 'Walkability', 'Peacefulness'],
  'gridcolor': 'rgb(172,172,172)',
  'linear': true,
});

$('#spidergraphcontainer').spidergraph('setactivedata', {'strokecolor': 'rgba(46,131,230,0.8)'...

$('#spidergraphcontainer').bind('spiderdatachange', function (ev, data) {
  processVibeTable(vibedata, {
    'honesty': data[0],
    'happiness': data[1],
    'liveliness': data[2],
    'walkability': data[3],
    'peacefulness': data[4],
  }, map);
});
});

```

For each point, the vibe magnitude is calculated as a weighted sum of the different vibe factors. For each weight, if the weight is decreased enough, it actually becomes negative – penalizing a listing for that vibe factor, if the user adjusts that vibe factor to be low enough. This enables the chart to have higher contrast, making it more clear which areas on the map best fit the user’s preferences.

```

function calcVibe(datapoint, weights) {
  return 2 * (weights['honesty'] - (defaultWeights['honesty'] * 4.0 / 5)) * datapoint['honesty'] +
    0.5 * (weights['happiness'] - (defaultWeights['happiness'] * 4.0 / 5)) * datapoint['happiness'] +
    2 * (weights['liveliness'] - (defaultWeights['liveliness'] * 4.0 / 5)) * datapoint['liveliness'] +
    2 * (weights['walkability'] - (defaultWeights['walkability'] * 4.0 / 5)) * datapoint['walkability'] +
    2 * (weights['peacefulness'] - (defaultWeights['peacefulness'] * 1.0 / 5)) * datapoint['peacefulness'];
}

```

## Societal Value

As mentioned before, the value our tool provides is the ability users have to make informed decisions about where to live and explore in Boston. Our tool provides insights that not even native Bostonians could produce since our tool analyzes hundreds of thousands of Airbnb listings and reviews which have been written not only by Bostonians but from those outside of Boston as well. Our tool provides a custom heatmap for each individual which can allow not only visitors, but native Bostonians to learn more about their city from an outsider’s perspective.

## Future Work

Future work on this project would include adding more vibe factors, such as culture, and allowing users to programmatically add vibe factors of their choice. Given the amount of data

available from Airbnb, it would be interesting to display the change in vibe over time. Airbnb has locations all over the world, and so expanding to create a global vibe map would be exciting. Additionally, we'd like to integrate data from sources other than just Airbnb (i.e. Twitter, Reddit, etc.). These other sources would provide us a more comprehensive set of data regarding the vibe of Boston and sample certain sections of the Boston population that may be missing from the Airbnb data alone. Finally, rewriting the data processing portion using a faster language such as Golang or Rust would enable user-side vibe computation, which would open the door to extensibility and greater scale.

## Conclusion

With this project, we hope to have shed light on the intangible aspects of living in Boston, that cannot be found through reviews on Google Maps or "How to visit Boston" guides. We were able to analyze and present findings from Airbnb in a directly actionable format, guiding people's decisions on where to stay in Boston.

We wished to abstract over the terminology of Airbnb's listings, and provide meaningful information about Boston itself. We hope that this tool will be useful not just to potential visitors of Boston, but to students and residents of Boston, helping them better understand the complex dynamics of a great city.

**Github repository:** <https://github.com/mbacvanski/vibe-of-boston>

**Live demo:** <https://mbacvanski.github.io/vibe-of-boston/>