**Identifying Prospective Areas in Denver for New Restaurants**

David Schneider

May 25, 2019

1. Introduction

1.1 Problem

Use publicly available data to guess to the best of our ability the answer to the question: “if someone is looking to open a restaurant in Denver, where would you recommend that they open it?”

1.2 Background

Every day, many people have hopes of opening their own restaurant. Some have done it before and are looking to repeat the process, and some are just trying it for the first time. Regardless, it would benefit anyone to do some guided research of where (in addition to how, but that is outside our scope) they should open this hypothetical restaurant instead of just picking a spot at random. Since we do not have data regarding the relative financial success of various restaurants in the area, we will have to make an educated guess on what we should look for when searching for a location to open a new restaurant. Some reasonable factors to consider might be:

* Total non-restaurant venues nearby (positive, indicates a busy area)
* Total other restaurants nearby (negative, potential for lost customers)
* Median income of neighborhood (positive, disposable income leads to eating out)
* Proximity to public transportation (positive, ease of access)

We also need to frame the scale that we will be analyzing this problem at. Due to data availability, we will use zip/postalcode as our target dimension. Due to scope of the model, a centroid point will be used for each postalcode to estimate lat/long coordinates.

1.3 Interest

The model might be useful to anyone interested in opening a restaurant in the Denver area. This could include individuals starting their first restaurant, current restaurant franchises looking to expand, or entrepreneurs looking to invest.

2. Data Acquisition and Cleaning

Data will be used for exploratory visualization as well as features for unsupervised machine learning. These raw numbers will be normalized for model input and weighted against the relative population of each zipcode.

2.1 Datasets / Sources

* List of zipcodes in Denver Metro Area: [source](https://www.zipcodestogo.com/Colorado/)(1)
* Venues in each zipcode: source- [FourSquare API](https://developer.foursquare.com/)(2)
  + Will be used for:
    - Total venues in zipcode
    - Type of each
* Median income data, flattened by zipcode: [source](https://www.irs.gov/pub/irs-soi/16zp06co.xls)(3)
* List of public transportation stations: [source](https://opendata.arcgis.com/datasets/17749050721d427399ab4e038028929d_6.csv)(4)
* Population data: [source](https://s3.amazonaws.com/SplitwiseBlogJB/2010+Census+Population+By+Zipcode+(ZCTA).csv)(5)

Only zipcodes which matched across all data sources were included. Most notably, a few zipcodes were dropped because there was no available latitude/longitude data, so there could not be any venue information for them queried via FourSquare.

2.2 Data Cleaning

The data for this project came from a variety of sources. First, a master list of all zipcodes in the Denver area of interest was created. For each source, the data was first aggregated to a zipcode level, then merged to the original list of zipcodes to append the new features. Most data sources contained entirely or nearly complete records for all zipcodes of interest. For those features where it was reasonable to make assumptions for missing data, replacement was used. For less obvious features such as income, the zipcodes were dropped from consideration. The effect of this was not large (*n*=4).

For modeling, features were standardized using a MinMaxScaler to set values between 0 and 1. Categorical features were also converted to either ordinal or nominal quantitative values based on context and scaled as well. Before performing scaling, outliers were first removed, then added back in after scales were created and assigned either the minimum or maximum value depending on context. The effect of this was small (*n*=1).

2.3 Feature Selection

Features were selected based on any data that I could find that was publicly available, could be aggregated to a zipcode level, for the Denver area, and might represent the attributes of an area that lend to it fostering successful restaurants. Because we only have a few features and they are all conceptually distinct, we will assume for the purposes of this exercise that they are sufficiently independent from one another. While it would be compelling to pursue the model with more features, more advanced statistical testing would need to be done to rule out the increasing likelihood that introducing more variables would increase the likelihood of there being overlap between their explanatory effect.

3. Exploratory Data Analysis

An initial data exploration phase was conducted in which collected features were plotted both independently and in conjunction with one another using Folium overlays. An initial map was first constructed to simply plot the known restaurants in the Denver area with markers. This map was then modified for the context of our analysis by overlaying and marking the boundaries for each zipcode, approximated by ZCTA. I then conducted an unvalidated preliminary exploration by investigating features created earlier in the data collection phase. If we had more exact quantitative data to test our model result, I would pursue more extensive statistical validation of the significance and independence of the features I chose to investigate. Regardless, using this method I was able to make some interesting identifications of areas with relatively wealthy populations that aren’t heavily populated with restaurants already, which seems like a good common-sense starting point. The advantage our map is able to provide is that it provides more context to this data—for example, I used shading to indicate restaurant density, which can provide more subtlety than simple text for “High” versus “Low”.

Conducting the exploratory data analysis mostly using Folium proved beneficial in this use case, mainly because the items of interest were themselves spatial data points. This provided the inherent advantage that once I layered the zipcode boundaries over the default map, any feature I visualized was being done so in the context of our variable of interest. This also enables easy creation and understanding of grouped data, such as total number of available public transportation (Lightrail) stations available for a given zipcode. This data exploration phase could be improved in the future by adding a step to include these aggregated statistics in the pop-up information for each zipcode on the Folium map. The maps created in this stage can be viewed on my github site: <https://github.com/davidschneider04/Coursera_Capstone>

Data exploration should again be conducted after the completion of the model to begin the iterative development process again. We visualize the model output for interesting relationships in the data and whether we can tweak its parameters to improve its accuracy. In this case, we plot the cluster labels created by the original model. We can save this plot, then change the model and compare the new plot it creates to the old one to see how minor changes upstream in the model affect its output. This highlights the axiom that in data science, one can never be truly “done”, as a model can never be perfect, but rather cyclically developed with incremental improvements each time.

4. Cluster Modeling

Due to data availability and format as well as the established context for the problem we are trying to solve, I decided to employ a cluster modeling method to further analyze the data. Because I was most interested in finding similarities between abstract groupings of data points, I decided on implementing a K-means model.

4.1 K-means

4.1.1 Formatting data (removing outliers, etc.)

Because K-means is influenced by the values of each individual data point used to calculate the error within the clusters, it will be sensitive to extreme outliers within the data that could cause oddly shaped clusters that don’t appropriately represent the features we are trying to model. As such I scanned the dataset for outliers to remove before normalization of the features and subsequent construction of the model, of which I found one (*n*=1). The value for the anomalous feature (“Restaurant Density”) was then set to the appropriate minimum/maximum (maximum in this case) value of the normalized feature for the filtered dataset.

Features from the dataset initially used for Folium mapping that would not be helpful for the model such as location data and plotting metadata were also dropped and the result copied to a new dataset which would be used to create the model. Data was then further reformatted to scale features with string values to be consumable by the modeling process. This flattened dataset was at this point indexed by zipcode, so we first make sure the data is properly sorted, then drop the index so that our dataset is composed only of features to be used for modeling.

4.1.2 Determination of Optimal *k*

The K-means model will create *k* total distinct clusters and we can choose any number for *k*. To decide on the best value, I used the “elbow method”, which essentially says to observe *k* in relation to how it impacts the average intracluster variation, ie. the mean of each datapoint’s distance from it to the centroid of its cluster. Since increasing *k* will naturally decrease this distance regardless, we look for *k* where the variation slows the relative rate of its decrease. For this dataset I decided on *k*=4. Here is the “elbow plot” I used to arrive at this decision:



4.1.3 Implementation of Model

Using the formatted and scaled dataset described above, I used the KMeans class from the SKLearn.Cluster package. The parameters for model construction were *k*=4 (determined above) and a *random\_state*=0, the modeling object was then fit to the formatted dataset. If future resources were to be allocated to this modeling exercise, I would run the model multiple times with different random\_state values and track/average the results. Fitting this model then results in cluster labels for each datapoint, which we can then merge back into our original, indexed dataset.

4.1.4 Visualization of Model

After merging the cluster labels created in the modeling process with the original, indexed dataset, we can create an arbitrary color scheme for the distinct clusters so that they can be visualized in the context of the map we created earlier during data exploration. We add location data back in and plot the zip code boundaries as Folium layers again, with the cluster determining the fill color for the polygons. This provides a distinctive way to see where our clusters lie in relation to each other, as long as we remember to provide a legend to help the user understand our coloring system.

5. Results

The results of the model do/do not support my initial self-guided hypothesis that 80206 would be clustered similarly to 80211. However, it does highlight several (n=22) other zipcodes which are modeled to be similar to 80211 based on the considered features. However, these are mostly located far away from the population epicenters of Denver, so taking into account our business heuristics of wanting to be relatively near the city center, we are left with three notable candidates: 80205, 80207, and 80220. For details on the granular data and transformations thereupon made to arrive at these modeled suggestions, please refer to the Detailed Methodology Appendix section. It also outlines the decision processes and explains why particular techniques were used over others. For a more interactive and likely better formatted experience, please use this Jupyter notebook: https://github.com/davidschneider04/Coursera\_Capstone/blob/master/CapstoneProjectTheBattleOfNeighborhoods.ipynb

6. Discussion/Conclusion

In this analysis, I compiled, visualized, and modeled various data regarding the Denver area and used it to create a cluster model using an unsupervised K-means algorithm. The use of an unsupervised cluster model was ideal because it allowed the significance of features to present themselves to us within a dataset we were relatively unfamiliar with. At this point, because we lack a test dataset with results for which restaurants in the area are more successful than others, it is difficult to draw particularly specific conclusions. Instead, what we do is identify markers of success and use those to initially guide our decision process. Using what we know about areas that are doing particularly well in popular opinion(9), we can examine other areas that are similar under the assumption that said similarity would hopefully lead to replicated success. This narrowed list of candidates can then be more meticulously researched by potential restauranteurs or investors to decide on a final location(s). From there, further research would be necessary to make a more exact location decision. I would suggest creating an entirely different model for this decision phase to predict the success of an establishment within a given zipcode given what we already know about that location’s attributes.

7. Future Directions

This model was heavily limited by a lack of data available for testing. This is not surprising, since the variable we are trying to predict, the potential success of a restaurant, is not easily measured, and even if it were it is not often publicly available. This situation of not having the “ground truth” as referenced in the course(8) is not uncommon and lends itself well to unsupervised learning models. For example, total profit may be high, but what if a venue has excessive outstanding loans? Is the venue from an existing franchise with an established history of success, or is it a potentially trendy but also risky mom-and-pop operation? Weighing these factors into one target metric is likely extremely complicated, but if someone with more financial experience than me were to create a dataset of these estimates, I could start working towards more quantitative predictive models, such as regression. Another interesting take could be finding a dataset of recent restaurant closures to indicate unsuccessful restaurants, and restaurants that have been open longer than a certain amount of time to indicate successful ones. Coding success as 1 and failure as 0, I could create a logistic regression model and refine it by iteratively minimizing the log-loss.

This model also only utilized a few features, which may be oversimplifying the factors at play in a restaurant’s success. If I had a more appropriate test dataset, I would also perform statistical validation to confirm that the features are both significant and not overly correlated with each other.

It would also be worthwhile exploring segmenting on dimensions besides postalcode. When I initially decided to evaluate based on postalcode, it was mostly due to the availability of income data as a good starting point. I was unaware of the peculiarities of said data-- postalcodes are actually defined by the delivery routes of the USPS, which of course are subject to frequent change. This means that they are also not polygons, so they are difficult to map. The commonly accepted solution is to use “ZCTAs” instead, which are approximations(6). More research could be done to determine a more appropriate zoning method such as city blocks or statistical neighborhood zones(7), but this would require a significant amount of effort since we would have to standardize the other datasets (income, etc.) to and as such is outside of the scope of this exercise.

Appendix A: Detailed Methodology (From Jupyter Notebook)

(Can be viewed interactively and with better formatting at https://github.com/davidschneider04/Coursera\_Capstone/blob/master/CapstoneProjectTheBattleOfNeighborhoods.htm)

Introduction[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Introduction)

# Problem Description:

### If someone is looking to open a restaurant in DENVER, where would you recommend that they open it?

# Problem Description:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Problem-Description:)

### If someone is looking to open a restaurant in DENVER, where would you recommend that they open it?[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#If-someone-is-looking-to-open-a-restaurant-in-DENVER,-where-would-you-recommend-that-they-open-it?)

# Background Discussion:

Since we do not have data regarding the relative financial success of various restaurants in the area, we will have to make an educated guess on what we should look for when searching for a location to open a new restaurant. Some reasonable factors to consider might be:

<ul><li>Total non-restaurant venues nearby (positive, indicates a busy area)</li>

   <li>Total other restaurants nearby (negative, potential for lost customers)</li>

   <li>Median income of neighborhood (positive, disposable income leads to eating out)</li>

   <li>Proximity to public transportation (positive, ease of access)</li>

   </ul>

​

We also need to frame the scale that we will be analyzing this problem at. Due to data availability, we will use zip/postalcode as our target dimension. Due to scope of the model, a centroid point will be used for each postalcode to estimate lat/long coordinates.

# Background Discussion:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Background-Discussion:)

Since we do not have data regarding the relative financial success of various restaurants in the area, we will have to make an educated guess on what we should look for when searching for a location to open a new restaurant. Some reasonable factors to consider might be:

* Total non-restaurant venues nearby (positive, indicates a busy area)
* Total other restaurants nearby (negative, potential for lost customers)
* Median income of neighborhood (positive, disposable income leads to eating out)
* Proximity to public transportation (positive, ease of access)

We also need to frame the scale that we will be analyzing this problem at. Due to data availability, we will use zip/postalcode as our target dimension. Due to scope of the model, a centroid point will be used for each postalcode to estimate lat/long coordinates.

xxxxxxxxxx

# Who Would Be Interested In This Project?:

The model might be useful to anyone interested in opening a restaurant in the Denver area. This could include individuals starting their first restaurant, current restaurant franchises looking to expand, or entrepreneurs looking to invest.

# Who Would Be Interested In This Project?:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Who-Would-Be-Interested-In-This-Project?:)

The model might be useful to anyone interested in opening a restaurant in the Denver area. This could include individuals starting their first restaurant, current restaurant franchises looking to expand, or entrepreneurs looking to invest.

# Data Description:

Will be used for exploratory visualization as well as features for unsupervised machine learning. These raw numbers will be normalized for model input and weighted against the relative population of each zipcode.

​

<ul>

   <li>List of zipcodes in Denver Metro Area <a href="https://www.zipcodestogo.com/Colorado/">source</a></li>

   <li>Venues in each zipcode (source: FourSquare API)<li>

   <ul>Will be used for:

       <li>Total venues in radius</li>

       <li>Type of each</li>

   </ul>

   <li>Median income data, flattened by zipcode <a href="https://www.irs.gov/pub/irs-soi/16zp06co.xls">source</a></li>

   <li>List of public transportation stations <a href="https://opendata.arcgis.com/datasets/17749050721d427399ab4e038028929d\_6.csv">source</a></li>

   <li>Population data <a href="https://s3.amazonaws.com/SplitwiseBlogJB/2010+Census+Population+By+Zipcode+(ZCTA).csv">source</a></li> </ul>

# Data Description:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Data-Description:)

Will be used for exploratory visualization as well as features for unsupervised machine learning. These raw numbers will be normalized for model input and weighted against the relative population of each zipcode.

* List of zipcodes in Denver Metro Area [source](https://www.zipcodestogo.com/Colorado/)
* Venues in each zipcode (source: FourSquare API)
* Will be used for:
  + Total venues in radius
  + Type of each
* Median income data, flattened by zipcode [source](https://www.irs.gov/pub/irs-soi/16zp06co.xls)
* List of public transportation stations [source](https://opendata.arcgis.com/datasets/17749050721d427399ab4e038028929d_6.csv)
* Population data [source](https://s3.amazonaws.com/SplitwiseBlogJB/2010+Census+Population+By+Zipcode+(ZCTA).csv)

xxxxxxxxxx

# Methodology:

# Methodology:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Methodology:)

In [1]:

#check if user has old python

import sys, re

assert re.match('[\d\.]{3}',sys.version).group(0) > '3.6', "Upgrade your python to one that is able to create f-strings (3.6 or greater) or else this code will error"

assert re.match('[\d\.]{3}',sys.version).group(0) > '3.6', "Upgrade your python to one that is able to create f-strings (3.6 or greater) or else this code will error"

. . .

#### Imports

#### Imports[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Imports)

In [2]:

import folium, re, geocoder, requests, json, uszipcode, sklearn, branca

import pandas as pd, numpy as np, matplotlib as mpl, matplotlib.pyplot as plt

import matplotlib.cm as cm, matplotlib.colors as colors

from pandas.io.json import json\_normalize

from geopy.geocoders import Nominatim

from sklearn.cluster import KMeans

from scipy.spatial.distance import cdist

. . .

#### Foursquare credentials

#### Foursquare credentials[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Foursquare-credentials)

In [3]:

with open('/Users/kutch/Desktop/IBM/foursquarecreds.txt') as f:

fsq = f.read()

client\_id = re.search("(?<=CLIENT\_ID)[ =]+'([A-Z0-9]+)", fsq).group(1)

client\_secret = re.search("(?<=CLIENT\_SECRET)[ =]+'([A-Z0-9]+)", fsq).group(1)

#not sure if coursera wants me to update this or not

VERSION = '20190501' # Foursquare API version

#from foursquare, function that extracts the category of the venue

def get\_category\_type(row):

try:

categories\_list = row['categories']

except:

categories\_list = row['venue.categories']

if len(categories\_list) == 0:

return None

else:

return categories\_list[0]['name']

. . .

#### Get all zipcodes in city of Denver

#### Get all zipcodes in city of Denver[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Get-all-zipcodes-in-city-of-Denver)

In [4]:

x

zipsearch = uszipcode.SearchEngine(simple\_zipcode=True)

all\_zips = pd.read\_html('https://www.zipcodestogo.com/Colorado/',

match='Zip Codes',

skiprows=range(3))[0]

all\_zips = all\_zips.drop(range(2,len(all\_zips.columns)),axis=1)

all\_zips.columns = ['Postalcode','City']

all\_zips = all\_zips.set\_index('Postalcode')

denver\_zips = all\_zips[all\_zips['City']=='Denver']

. . .

#### Append lat/long to zipcodes

#### Append lat/long to zipcodes[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Append-lat/long-to-zipcodes)

In [5]:

ll\_data = [[zipcode, zipsearch.by\_zipcode(zipcode).lat, zipsearch.by\_zipcode(zipcode).lng] for zipcode in denver\_zips.index]

ll\_df = pd.DataFrame(data=ll\_data, columns=['Postalcode','Latitude','Longitude']).dropna().set\_index('Postalcode')

denver\_zips = denver\_zips.merge(ll\_df,left\_index=True,right\_index=True)

ll\_data = [[zipcode, zipsearch.by\_zipcode(zipcode).lat, zipsearch.by\_zipcode(zipcode).lng] for zipcode in denver\_zips.index]

. . .

x

#### Get population data

Important note: Census data cannot be mapped exactly to zip code (<a href="https://www.quora.com/Where-can-I-find-U-S-Census-data-with-population-per-ZIP-code-Other-details-such-as-age-gender-breakdown-would-be-helpful">see more</a>)

#### Get population data[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Get-population-data)

Important note: Census data cannot be mapped exactly to zip code ([see more](https://www.quora.com/Where-can-I-find-U-S-Census-data-with-population-per-ZIP-code-Other-details-such-as-age-gender-breakdown-would-be-helpful))

In [6]:

population = pd.read\_csv("https://s3.amazonaws.com/SplitwiseBlogJB/2010+Census+Population+By+Zipcode+(ZCTA).csv")

population = population.rename(mapper={population.columns[0]:'Postalcode',

population.columns[1]:'Population'}, axis=1)

population = population.set\_index('Postalcode')

population\_denver = population.merge(denver\_zips, left\_index=True, right\_index=True).drop(denver\_zips.columns, axis=1)

. . .

#### Get income data

#### Get income data[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Get-income-data)

In [7]:

income = pd.read\_excel('https://www.irs.gov/pub/irs-soi/16zp06co.xls',

header=None,

indexcol=0,

skiprows=range(6),

usecols=range(3)).dropna()

income.columns = ['Postalcode','Income','NumPeople']

income = income[income['Income']!='Total']

for index, row in income.iterrows():

if re.match('\.',str(row['NumPeople'])):

income = income.drop(index)

income['NumPeople'] = income['NumPeople'].astype(float)

income = income.set\_index('Postalcode')

#most common category

tgb = income.drop(['Income'],axis=1).groupby(by=['Postalcode']).max()

income = tgb.merge(income, on=['Postalcode'], suffixes=['\_x',''])

income = income[income['NumPeople\_x']==income['NumPeople']].drop(['NumPeople\_x','NumPeople'], axis=1)

income\_denver = income.merge(denver\_zips, left\_index=True, right\_index=True).drop(denver\_zips.columns, axis=1)

income\_denver = income\_denver.rename(mapper={'Income':'MedianIncomeBracket'},axis=1)

income\_denver['MedianIncomeBracket'] = income\_denver['MedianIncomeBracket'].astype('str')

income\_denver = income\_denver.replace({'under':'\ through\ '},regex=True)

income\_denver = income\_denver.groupby(by=income\_denver.index).min()

. . .

#### Get public transportation data

#### Get public transportation data[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Get-public-transportation-data)

In [8]:

lightrail = pd.read\_csv('https://opendata.arcgis.com/datasets/17749050721d427399ab4e038028929d\_6.csv')

lightrail = lightrail.rename(mapper={'ZIPCODE':'Postalcode', 'PID':'NumLightrailStations'},axis=1)

lightrail = pd.DataFrame(lightrail.groupby(by=['Postalcode']).count()['NumLightrailStations'])

. . .

#### Get venue data

#### Get venue data[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Get-venue-data)

In [9]:

x

#for a given list of zipcodes

def get\_nearby\_venues(zipcodes, latitudes, longitudes):

radius, limit = 500, 100

global client\_id, client\_secret

venues\_list=[]

for name, lat, lng in zip(zipcodes, latitudes, longitudes):

# create the API request URL

url = f'https://api.foursquare.com/v2/venues/explore?&client\_id={client\_id}&client\_secret={client\_secret}&v={VERSION}&ll={lat},{lng}&radius={radius}&limit={limit}'

# make the GET request

results = requests.get(url).json()["response"]['groups'][0]['items']

# return only relevant information for each nearby venue

venues\_list.append([(

name,

lat,

lng,

v['venue']['name'],

v['venue']['location']['lat'],

v['venue']['location']['lng'],

v['venue']['categories'][0]['name']) for v in results])

nearby\_venues = pd.DataFrame([item for venue\_list in venues\_list for item in venue\_list])

nearby\_venues.columns = ['Zipcode',

'ZipcodeLatitude',

'ZipcodeLongitude',

'Venue',

'VenueLatitude',

'VenueLongitude',

'VenueCategory']

return(nearby\_venues)

​

#get venue info for all denver zipcodes

venues\_denver = get\_nearby\_venues(zipcodes=pd.Series(denver\_zips.index),

latitudes=denver\_zips['Latitude'],

longitudes=denver\_zips['Longitude'])

venues\_denver = venues\_denver.rename(mapper={'Zipcode':'Postalcode'},axis=1)

venues\_denver = get\_nearby\_venues(zipcodes=pd.Series(denver\_zips.index),

. . .

In [10]:

restaurant\_categories = pd.read\_csv("https://raw.githubusercontent.com/davidschneider04/Coursera\_Capstone/master/RestaurantCategories.csv"

,encoding = 'latin-1')

restaurants\_denver = venues\_denver.merge(restaurant\_categories, on='VenueCategory')

restaurants\_denver = restaurants\_denver.drop(['ZipcodeLatitude','ZipcodeLongitude'],axis=1)

restaurants\_denver = restaurants\_denver[['Postalcode','VenueCategory','Venue','VenueLatitude','VenueLongitude']]

. . .

### First, let's get a general sense of the distribution of restauarants in the area using Folium.

### First, let's get a general sense of the distribution of restauarants in the area using Folium.[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#First,-let's-get-a-general-sense-of-the-distribution-of-restauarants-in-the-area-using-Folium.)

#### Create a map centered on Denver

#### Create a map centered on Denver[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Create-a-map-centered-on-Denver)

In [11]:

def create\_map\_denver():

location = Nominatim(user\_agent="den\_explorer").geocode("Denver, CO")

latitude, longitude = location.latitude, location.longitude

mapobj = folium.Map(location=[latitude, longitude], zoom\_start=12)

return mapobj

map\_denver = create\_map\_denver()

. . .

xxxxxxxxxx

#### Frame the zipcodes we are interested in. Information originally sourced from:

https://opendata.arcgis.com/datasets/6b6091f299204e4c9c406a624baf43e6\_10.geojson

#### Frame the zipcodes we are interested in. Information originally sourced from:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Frame-the-zipcodes-we-are-interested-in.-Information-originally-sourced-from:)

<https://opendata.arcgis.com/datasets/6b6091f299204e4c9c406a624baf43e6_10.geojson>

In [12]:

x

with open('/Users/kutch/Desktop/IBM/Colorado\_ZIP\_Code\_Tabulation\_Areas\_ZCTA.geojson', 'r') as gjson:

data = json.load(gjson)

tmp = data

dzips = list(denver\_zips.index.astype('str').unique())

map\_boundaries = {'type': 'FeatureCollection', 'features': []}

for item in tmp['features']:

item['properties']['name'] = str(item['properties']['ZCTA5CE10'])

zipcode = item['properties']['ZCTA5CE10']

if zipcode in dzips:

map\_boundaries['features'].append(item)

tmp = data

. . .

In [13]:

def style\_function(feature):

return {

'fillOpacity': 0.5,

'weight': 1}

​

for feature in map\_boundaries['features']:

i=1

geojson = folium.GeoJson(

feature,

name=f'mapfeature{i}',

style\_function=style\_function

).add\_to(map\_denver)

popup = folium.Popup(feature['properties']['name'])

popup.add\_to(geojson)

geojson.add\_to(map\_denver)

i+=1

. . .

#### Add restaurant markers to map

#### Add restaurant markers to map[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Add-restaurant-markers-to-map)

In [14]:

for lat, lng, label in zip(restaurants\_denver['VenueLatitude'],

restaurants\_denver['VenueLongitude'],

restaurants\_denver['Venue']):

label = folium.Popup(str(label), parse\_html=True)

restaurant\_marker = folium.CircleMarker(

[lat, lng],

radius=5,

popup=label,

color='blue',

fill=True,

fill\_color='#3186cc',

fill\_opacity=0.7,

parse\_html=False)

restaurant\_marker.add\_to(map\_denver)

. . .

#### Render the map

#### Render the map[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Render-the-map)

In [15]:

#click a circle marker to see the name of the restaurant,

##a general area to see the zipcode

map\_denver

Out[15]:

. . .

### Obviously, this is an incredibly complex problem and we could refine a model forever if the stakes here were higher and there were more resources available. For now, we will have to create a heuristic to simplify our process, our model, and its intepretation.

In this hypothetical situation, let's say our restaurant is relatively upscale. From personal experience, I would say an oversimplified picture of a successful environment for this theoretical venue would be a location where potential customers have enough money to eat out often at our restaurant, and few enough restaurants nearby where we will not be competing for customers constantly. So, we will search for zipcodes that satisfy the following criteria:

<ul><li>Low restaurant:person ratio</li>

   <li>High median income</li></ul>

### Obviously, this is an incredibly complex problem and we could refine a model forever if the stakes here were higher and there were more resources available. For now, we will have to create a heuristic to simplify our process, our model, and its intepretation.[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Obviously,-this-is-an-incredibly-complex-problem-and-we-could-refine-a-model-forever-if-the-stakes-here-were-higher-and-there-were-more-resources-available.-For-now,-we-will-have-to-create-a-heuristic-to-simplify-our-process,-our-model,-and-its-intepretation.)

In this hypothetical situation, let's say our restaurant is relatively upscale. From personal experience, I would say an oversimplified picture of a successful environment for this theoretical venue would be a location where potential customers have enough money to eat out often at our restaurant, and few enough restaurants nearby where we will not be competing for customers constantly. So, we will search for zipcodes that satisfy the following criteria:

* Low restaurant:person ratio
* High median income

xxxxxxxxxx

#### restaurant:person ratio

#### restaurant:person ratio[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#restaurant:person-ratio)

In [16]:

x

zipvenues\_denver = venues\_denver.set\_index('Postalcode').groupby(by=['Postalcode']).count()['VenueCategory']

zipvenues\_denver = pd.DataFrame(zipvenues\_denver).rename(mapper={'VenueCategory':'TotalVenues'}, axis=1)

#restaurant density

rd\_denver = zipvenues\_denver.merge(population\_denver,

left\_index=True, right\_index=True)

rd\_denver['restaurant\_density'] = rd\_denver['TotalVenues'] / rd\_denver['Population']

rd\_denver = rd\_denver.replace([np.inf], 0)

rd\_denver = rd\_denver.drop([col for col in rd\_denver if col != 'restaurant\_density'],

axis=1)

##remove outliers

tmp = rd\_denver

outliers = ['80202']

tmp = tmp.loc[[i for i in tmp.index if str(i) not in outliers]]

##scale

mms = sklearn.preprocessing.MinMaxScaler()

tmp['restaurant\_density'] = mms.fit\_transform(tmp)

rd\_denver = rd\_denver.merge(tmp, how='left',

left\_index=True, right\_index=True,

suffixes=('\_x',''))

rd\_denver = rd\_denver.drop(columns=[col for col in rd\_denver.columns if re.search('\\_x',col)])

##add outliers back in

rd\_denver = rd\_denver.replace([np.nan], 1)

. . .

#### Create a combined dataset of zipcode attributes

#### Create a combined dataset of zipcode attributes[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Create-a-combined-dataset-of-zipcode-attributes)

In [17]:

x

#datasets:

##denver\_zips

##income\_denver

##rd\_denver

##lightrail

denver\_data = denver\_zips

#append income, inner join

denver\_data = denver\_data.merge(income\_denver, left\_index=True, right\_index=True)

#append restaurant density, inner join

denver\_data = denver\_data.merge(rd\_denver, left\_index=True, right\_index=True)

#append lightrail station data, left join bc we can assume 0 for missing values

##scale values for modeling

mms = sklearn.preprocessing.MinMaxScaler()

lightrail['ScaledLightrailStations'] = mms.fit\_transform(lightrail)

lightrail = lightrail.drop(['NumLightrailStations'],axis=1)

denver\_data = denver\_data.merge(lightrail, how='left', left\_index=True, right\_index=True)

denver\_data['ScaledLightrailStations'] = denver\_data['ScaledLightrailStations'].replace([np.nan],0)

. . .

xxxxxxxxxx

### Let's refresh our map with this new data

#### Shade zipcode polygons according to inverse restaurant density

### Let's refresh our map with this new data[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Let's-refresh-our-map-with-this-new-data)

#### Shade zipcode polygons according to inverse restaurant density[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Shade-zipcode-polygons-according-to-inverse-restaurant-density)

In [18]:

map\_denver = create\_map\_denver()

denver\_data['restaurant\_density\_color'] = denver\_data['restaurant\_density'].astype('str')

denver\_data['restaurant\_density\_color'] = denver\_data['restaurant\_density\_color'].apply(mpl.colors.to\_hex)

. . .

In [19]:

x

def style\_function(feature):

global denver\_data

color = denver\_data.loc[int(feature['properties']['name'])]['restaurant\_density\_color']

return {

'fillColor': color,

'color': color,

'fillOpacity': 0.5,

'weight': 1}

. . .

#### Only plot polygons for high-income zipcodes

#### Only plot polygons for high-income zipcodes[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Only-plot-polygons-for-high-income-zipcodes)

In [20]:

i=1

for feature in map\_boundaries['features']:

zipcodeval = feature['properties']['name']

if int(zipcodeval) in denver\_data[~denver\_data['MedianIncomeBracket'].isin(['$1 \\ through\\ $25,000'])].index:

geojson = folium.GeoJson(

feature,

name=f'mapfeature{i}',

style\_function=style\_function

).add\_to(map\_denver)

popup = folium.Popup(zipcodeval)

popup.add\_to(geojson)

geojson.add\_to(map\_denver)

i+=1

name=f'mapfeature{i}',

. . .

In [21]:

map\_denver

Out[21]:

. . .

xxxxxxxxxx

#### It seems like the zipcode within the city proper that stands out the most is 80206. Let's see if we can confirm or deny our suspicions with an unsupervised machine learning model.

#### It seems like the zipcode within the city proper that stands out the most is 80206. Let's see if we can confirm or deny our suspicions with an unsupervised machine learning model.[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#It-seems-like-the-zipcode-within-the-city-proper-that-stands-out-the-most-is-80206.-Let's-see-if-we-can-confirm-or-deny-our-suspicions-with-an-unsupervised-machine-learning-model.)

### Similarity Modeling

Suppose we do some more research, and we learn that many restaurants in a certain portion of town ("Highlands") are thriving (<a href="https://www.westword.com/restaurants/lohi-now-home-to-over-75-restaurants-and-brs-11258151">source</a>). But, we then find this research to be largely outdated and the area currently overpriced. Can we find a suitable substitute? Absent other data, this neighborhood can be our estimate of a good area to open a restaurant in. We can then use similarity modeling to find other areas in Denver that share the same features which make it favorable for restaurant development, but at a hopefully lower price. A <b>K-Means</b> model will allow us to create arbitrary clusters to find other zipcodes that are similar to the thriving Highlands neighborhood based on a mixture of our features.

​

#### "LoHi" is a neighborhood, but we've created our dataset at a zipcode level, so we need to standardize.

Using <a href="https://www.denvergov.org/maps/map/neighborhoods">this map</a>, we can see that the "Highlands" neighborhood is a subset of the <b>80211</b> zipcode. This means we will take the 80211 row of our "denver\_data" DataFrame to represent a good spot for a restaurant going into our model.

### Similarity Modeling[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Similarity-Modeling)

Suppose we do some more research, and we learn that many restaurants in a certain portion of town ("Highlands") are thriving ([source](https://www.westword.com/restaurants/lohi-now-home-to-over-75-restaurants-and-brs-11258151)). But, we then find this research to be largely outdated and the area currently overpriced. Can we find a suitable substitute? Absent other data, this neighborhood can be our estimate of a good area to open a restaurant in. We can then use similarity modeling to find other areas in Denver that share the same features which make it favorable for restaurant development, but at a hopefully lower price. A **K-Means** model will allow us to create arbitrary clusters to find other zipcodes that are similar to the thriving Highlands neighborhood based on a mixture of our features.

#### "LoHi" is a neighborhood, but we've created our dataset at a zipcode level, so we need to standardize.[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#%22LoHi%22-is-a-neighborhood,-but-we've-created-our-dataset-at-a-zipcode-level,-so-we-need-to-standardize.)

Using [this map](https://www.denvergov.org/maps/map/neighborhoods), we can see that the "Highlands" neighborhood is a subset of the **80211** zipcode. This means we will take the 80211 row of our "denver\_data" DataFrame to represent a good spot for a restaurant going into our model.

xxxxxxxxxx

#### Create formatted dataset for modeling

#### Create formatted dataset for modeling[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Create-formatted-dataset-for-modeling)

In [22]:

#create a dataset more appropriate for modeling

denver\_data\_model = denver\_data

#not appropriate features

denver\_data\_model = denver\_data\_model.drop(['City','Latitude','Longitude','restaurant\_density\_color'], axis=1)

#reformat income data

##as ordinal

mapper = {'$1 \\ through\\ $25,000':1, '$25,000 \\ through\\ $50,000':2, '$100,000 \\ through\\ $200,000':3}

denver\_data\_model['MedianIncomeBracket'] = denver\_data\_model['MedianIncomeBracket'].replace(mapper)

##scaled

mms = sklearn.preprocessing.MinMaxScaler()

mi = pd.DataFrame(denver\_data\_model['MedianIncomeBracket'])

mi['ScaledMedianIncomeBracket'] = mms.fit\_transform(mi)

mi = mi.drop(['MedianIncomeBracket'], axis=1)

##combine

denver\_data\_model = denver\_data\_model.merge(mi, left\_index=True, right\_index=True)

denver\_data\_model = denver\_data\_model.drop(['MedianIncomeBracket'],axis=1)

. . .

xxxxxxxxxx

#### Let's inspect the dataset to ensure it's appropriate

#### Let's inspect the dataset to ensure it's appropriate[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Let's-inspect-the-dataset-to-ensure-it's-appropriate)

In [23]:

print(denver\_data\_model.shape)

denver\_data\_model.head()

(38, 3)

Out[23]:

|  | **restaurant\_density** | **ScaledLightrailStations** | **ScaledMedianIncomeBracket** |
| --- | --- | --- | --- |
| **Postalcode** |  |  |  |
| **80202** | 1.000000 | 0.555556 | 0.5 |
| **80203** | 0.834537 | 0.000000 | 0.5 |
| **80204** | 0.221874 | 1.000000 | 0.0 |
| **80205** | 0.298907 | 0.333333 | 0.0 |
| **80206** | 0.108227 | 0.000000 | 0.5 |

. . .

xxxxxxxxxx

#### Find optimal value for k

#### Find optimal value for k[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Find-optimal-value-for-k)

In [24]:

distortions = []

krange = range(1,10)

for k in krange:

kmeanModel = KMeans(n\_clusters=k).fit(denver\_data\_model)

kmeanModel.fit(denver\_data\_model)

distortions.append(sum(np.min(cdist(denver\_data\_model, kmeanModel.cluster\_centers\_, 'euclidean'), axis=1)) / denver\_data\_model.shape[0])

​

#elbow plot

plt.plot(krange, distortions, 'r.-')

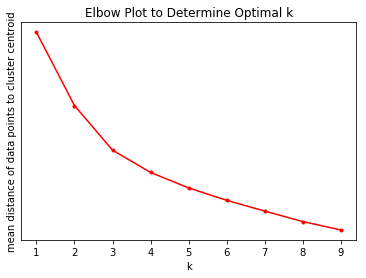
plt.xlabel('k')

plt.ylabel('mean distance of data points to cluster centroid')

plt.title('Elbow Plot to Determine Optimal k')

plt.yticks([])

plt.show()

****

. . .

xxxxxxxxxx

#### k=4

It seems like the slope of the line tangent to the curve of the mean point:centroid distance slows its decrease around k=4; we will choose this as the number of clusters to use when building the final k-means model

#### k=4[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#k=4)

It seems like the slope of the line tangent to the curve of the mean point:centroid distance slows its decrease around k=4; we will choose this as the number of clusters to use when building the final k-means model

#### K-means model

#### K-means model[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#K-means-model)

In [25]:

x

kval = 4

#sort the dataset

denver\_data\_model = denver\_data\_model.sort\_values(by=['restaurant\_density','ScaledLightrailStations','ScaledMedianIncomeBracket'],ascending=False)

denver\_data\_clustering = denver\_data\_model

denver\_data\_clustering = denver\_data\_clustering.reset\_index().drop(['Postalcode'], axis=1)

kmm = KMeans(n\_clusters=kval, random\_state=0).fit(denver\_data\_clustering)

. . .

xxxxxxxxxx

#### Insert cluster labels back and index at 1

#### Insert cluster labels back and index at 1[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Insert-cluster-labels-back-and-index-at-1)

In [26]:

denver\_data\_model['cluster\_label'] = [label+1 for label in kmm.labels\_]

. . .

#### Profile the clusters so we can give them descriptions

#### Profile the clusters so we can give them descriptions[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Profile-the-clusters-so-we-can-give-them-descriptions)

In [27]:

cluster\_info = denver\_data\_model.reset\_index()

cluster\_info = cluster\_info.drop(['Postalcode'], axis=1)

cluster\_info = cluster\_info.set\_index(['cluster\_label'])

cluster\_info = cluster\_info.groupby(cluster\_info.index).mean()

cluster\_info

Out[27]:

|  | **restaurant\_density** | **ScaledLightrailStations** | **ScaledMedianIncomeBracket** |
| --- | --- | --- | --- |
| **cluster\_label** |  |  |  |
| **1** | 0.221874 | 1.000000 | 0.000000 |
| **2** | 0.254269 | 0.027778 | 0.562500 |
| **3** | 0.128583 | 0.035354 | 0.000000 |
| **4** | 0.767900 | 0.126984 | 0.142857 |

. . .

#### We can subjectively judge the following and use the statistics to create labels:

<ol><li> Restaurants: medium, Lightrails: high, Income: low. <b>Label:</b> Accessible</li>

   <li> Restaurants: medium, Lightrails: low, Income: high <b>Label:</b> High Income</li>

   <li> Restaurants: low, Lightrails: low, Income: low <b>Label:</b> Underdeveloped</li>

   <li> Restaurants: high, Lightrails: medium, Income: medium. <b>Label:</b> Crowded</li>

   </ol>

#### We can subjectively judge the following and use the statistics to create labels:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#We-can-subjectively-judge-the-following-and-use-the-statistics-to-create-labels:)

1. Restaurants: medium, Lightrails: high, Income: low. **Label:** Accessible
2. Restaurants: medium, Lightrails: low, Income: high **Label:** High Income
3. Restaurants: low, Lightrails: low, Income: low **Label:** Underdeveloped
4. Restaurants: high, Lightrails: medium, Income: medium. **Label:** Crowded

In [28]:

#assign the labels

mapper = {1:'Accessible', 2:'High Income', 3:'Underdeveloped', 4:'Crowded'}

denver\_data\_model['cluster\_label'] = denver\_data\_model['cluster\_label'].replace(mapper)

#assign the labels

. . .

#### Add location data back in so we can visualize the clusters

#### Add location data back in so we can visualize the clusters[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Add-location-data-back-in-so-we-can-visualize-the-clusters)

In [29]:

denver\_data\_modelplot = denver\_data\_model.merge(denver\_data,

left\_index=True, right\_index=True

,suffixes=('\_x',''))

. . .

In [30]:

x

denver\_data\_modelplot = denver\_data\_modelplot.drop([col for col in denver\_data\_modelplot.columns if re.search('(\\_x)|(color)', col)]

,axis=1)

denver\_data\_modelplot = denver\_data\_modelplot.drop(['MedianIncomeBracket']

,axis=1)

,axis=1)

. . .

x

#### Set color scheme for the clusters

#### Set color scheme for the clusters[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Set-color-scheme-for-the-clusters)

In [31]:

x = np.arange(kval)

ys = [i + x + (i\*x)\*\*2 for i in range(kval)]

colors\_array = cm.rainbow(np.linspace(0, 1, len(ys)))

rainbow = [colors.rgb2hex(i) for i in colors\_array]

rainbowdict = {i:color for (i, color) in enumerate(rainbow)}

clustermapper = {value: key for key, value in mapper.items()}

clustermapper = {value: key for key, value in mapper.items()}

. . .

In [32]:

denver\_data\_modelplot['cluster\_code'] = denver\_data\_modelplot['cluster\_label'].map(clustermapper)-1

denver\_data\_modelplot['cluster\_color'] = denver\_data\_modelplot['cluster\_code'].map(rainbowdict)

. . .

#### Create a new map to show clusters

#### Create a new map to show clusters[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Create-a-new-map-to-show-clusters)

In [33]:

x

map\_denver = create\_map\_denver()

. . .

In [34]:

x

def style\_function(feature):

global denver\_data\_modelplot

color = denver\_data\_modelplot.loc[int(feature['properties']['name'])]['cluster\_color']

return {

'fillColor': color,

'color': color,

'fillOpacity': 0.5,

'weight': 1}

​

i=1

for feature in map\_boundaries['features']:

zipcodeval = feature['properties']['name']

if int(zipcodeval) in list(denver\_data\_modelplot.index):

geojson = folium.GeoJson(

feature,

name=f'mapfeature{i}',

style\_function=style\_function

).add\_to(map\_denver)

popup = folium.Popup(zipcodeval)

popup.add\_to(geojson)

geojson.add\_to(map\_denver)

i+=1

popup.add\_to(geojson)

. . .

xxxxxxxxxx

#### Create a legend

#### Create a legend[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Create-a-legend)

In [35]:

legend\_html\_base = f"""

<div style="position: fixed;

bottom: 50px; left: 50px; width: 150px; height: 125px;

border:2px solid grey; z-index:9999; font-size:12px; background:white;

">&nbsp; <b>Legend</b> <br>

&nbsp;"""

for cluster in list(denver\_data\_modelplot['cluster\_label'].unique()):

color = denver\_data\_modelplot[denver\_data\_modelplot['cluster\_label']==cluster]['cluster\_color'].iloc[0]

legend\_html\_base += f"""{cluster} &nbsp; <i class="fa fa-map-marker fa-2x" style="color:{color}"></i><br> &nbsp;"""

legend\_html = legend\_html\_base + """</div>"""

map\_denver.get\_root().html.add\_child(folium.Element(legend\_html))

legend\_html\_base += f"""{cluster} &nbsp; <i class="fa fa-map-marker fa-2x" style="color:{color}"></i><br> &nbsp;"""

Out[35]:

<branca.element.Element at 0x1249ee5c0>

. . .

#### Render the map

#### Render the map[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Render-the-map)

In [36]:

map\_denver

Out[36]:

. . .

xxxxxxxxxx

#### While the colors of our clusters on the map are arbitrary, it is a happy coincidence for implicit understanding that red represents the color of the presumably least desirable cluster-- "Crowded" (with other restaruants)

#### While the colors of our clusters on the map are arbitrary, it is a happy coincidence for implicit understanding that red represents the color of the presumably least desirable cluster-- "Crowded" (with other restaruants)[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#While-the-colors-of-our-clusters-on-the-map-are-arbitrary,-it-is-a-happy-coincidence-for-implicit-understanding-that-red-represents-the-color-of-the-presumably-least-desirable-cluster---%22Crowded%22-(with-other-restaruants))

#### Create the map again, but this time only with zipcodes matching the cluster for 80211.

#### Create the map again, but this time only with zipcodes matching the cluster for 80211.[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Create-the-map-again,-but-this-time-only-with-zipcodes-matching-the-cluster-for-80211.)

In [37]:

map\_denver = create\_map\_denver()

targetcluster = denver\_data\_modelplot.loc[80211]['cluster\_label']

i=1

for feature in map\_boundaries['features']:

zipcodeval = int(feature['properties']['name'])

if zipcodeval in list(denver\_data\_modelplot.index) and denver\_data\_modelplot.loc[zipcodeval]['cluster\_label'] == targetcluster:

geojson = folium.GeoJson(

feature,

name=f'mapfeature{i}',

style\_function=style\_function

).add\_to(map\_denver)

popup = folium.Popup(str(zipcodeval))

popup.add\_to(geojson)

geojson.add\_to(map\_denver)

i+=1

map\_denver

name=f'mapfeature{i}',

Out[37]:

. . .

#### For reference, here are all of the zipcodes within the cluster we are interested in:

#### For reference, here are all of the zipcodes within the cluster we are interested in:[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#For-reference,-here-are-all-of-the-zipcodes-within-the-cluster-we-are-interested-in:)

In [38]:

denver\_data\_modelplot[denver\_data\_modelplot['cluster\_label']==denver\_data\_modelplot.loc[80211]['cluster\_label']]

Out[38]:

|  | **ScaledMedianIncomeBracket** | **cluster\_label** | **City** | **Latitude** | **Longitude** | **restaurant\_density** | **ScaledLightrailStations** | **cluster\_code** | **cluster\_color** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Postalcode** |  |  |  |  |  |  |  |  |  |
| **80236** | 0.0 | Underdeveloped | Denver | 39.65 | -105.03 | 0.361155 | 0.000000 | 2 | #d4dd80 |
| **80211** | 0.0 | Underdeveloped | Denver | 39.76 | -105.02 | 0.331863 | 0.000000 | 2 | #d4dd80 |
| **80230** | 0.0 | Underdeveloped | Denver | 39.72 | -104.89 | 0.308131 | 0.000000 | 2 | #d4dd80 |
| **80205** | 0.0 | Underdeveloped | Denver | 39.76 | -104.97 | 0.298907 | 0.333333 | 2 | #d4dd80 |
| **80235** | 0.0 | Underdeveloped | Denver | 39.65 | -105.09 | 0.153821 | 0.000000 | 2 | #d4dd80 |
| **80219** | 0.0 | Underdeveloped | Denver | 39.71 | -105.03 | 0.151994 | 0.000000 | 2 | #d4dd80 |
| **80222** | 0.0 | Underdeveloped | Denver | 39.67 | -104.93 | 0.145043 | 0.111111 | 2 | #d4dd80 |
| **80247** | 0.0 | Underdeveloped | Denver | 39.69 | -104.88 | 0.137284 | 0.000000 | 2 | #d4dd80 |
| **80234** | 0.0 | Underdeveloped | Denver | 39.91 | -105.01 | 0.112251 | 0.000000 | 2 | #d4dd80 |
| **80223** | 0.0 | Underdeveloped | Denver | 39.69 | -105.00 | 0.097119 | 0.111111 | 2 | #d4dd80 |
| **80226** | 0.0 | Underdeveloped | Denver | 39.71 | -105.09 | 0.094130 | 0.000000 | 2 | #d4dd80 |
| **80231** | 0.0 | Underdeveloped | Denver | 39.67 | -104.89 | 0.090623 | 0.000000 | 2 | #d4dd80 |
| **80228** | 0.0 | Underdeveloped | Denver | 39.69 | -105.15 | 0.077283 | 0.000000 | 2 | #d4dd80 |
| **80221** | 0.0 | Underdeveloped | Denver | 39.82 | -105.01 | 0.077065 | 0.111111 | 2 | #d4dd80 |
| **80220** | 0.0 | Underdeveloped | Denver | 39.73 | -104.91 | 0.069467 | 0.000000 | 2 | #d4dd80 |
| **80227** | 0.0 | Underdeveloped | Denver | 39.66 | -105.10 | 0.069182 | 0.000000 | 2 | #d4dd80 |
| **80207** | 0.0 | Underdeveloped | Denver | 39.76 | -104.92 | 0.054892 | 0.000000 | 2 | #d4dd80 |
| **80216** | 0.0 | Underdeveloped | Denver | 39.79 | -104.95 | 0.053304 | 0.111111 | 2 | #d4dd80 |
| **80229** | 0.0 | Underdeveloped | Denver | 39.85 | -104.95 | 0.046961 | 0.000000 | 2 | #d4dd80 |
| **80239** | 0.0 | Underdeveloped | Denver | 39.78 | -104.84 | 0.043120 | 0.000000 | 2 | #d4dd80 |
| **80241** | 0.0 | Underdeveloped | Denver | 39.93 | -104.96 | 0.036684 | 0.000000 | 2 | #d4dd80 |
| **80260** | 0.0 | Underdeveloped | Denver | 39.87 | -105.01 | 0.018537 | 0.000000 | 2 | #d4dd80 |

. . .

xxxxxxxxxx

#### If we want, we can join back to the original venues dataframe to get a more granular look now that we have filtered the number of data points down to a more manageable amount and do more data exploration based on that. We will create this dataframe, but leave the investigation of specific places for another time.

#### If we want, we can join back to the original venues dataframe to get a more granular look now that we have filtered the number of data points down to a more manageable amount and do more data exploration based on that. We will create this dataframe, but leave the investigation of specific places for another time.[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#If-we-want,-we-can-join-back-to-the-original-venues-dataframe-to-get-a-more-granular-look-now-that-we-have-filtered-the-number-of-data-points-down-to-a-more-manageable-amount-and-do-more-data-exploration-based-on-that.-We-will-create-this-dataframe,-but-leave-the-investigation-of-specific-places-for-another-time.)

In [39]:

vdtmp = venues\_denver

vdtmp = vdtmp.set\_index('Postalcode')

potential\_competitors = denver\_data\_modelplot.merge(vdtmp,left\_index=True,right\_index=True)

#reorder columns, drop unnecessary columns

potential\_competitors = potential\_competitors[['VenueCategory','Venue','VenueLatitude','VenueLongitude'

,'ScaledMedianIncomeBracket','cluster\_label',

'restaurant\_density','ScaledLightrailStations']]

potential\_competitors = potential\_competitors.sort\_values(by=['Postalcode', 'VenueCategory','Venue'])

potential\_competitors.head()

,'ScaledMedianIncomeBracket','cluster\_label','restaurant\_density'''ScaledLightrailStations]]

Out[39]:

|  | **VenueCategory** | **Venue** | **VenueLatitude** | **VenueLongitude** | **ScaledMedianIncomeBracket** | **cluster\_label** | **restaurant\_density** | **ScaledLightrailStations** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Postalcode** |  |  |  |  |  |  |  |  |
| **80202** | Accessories Store | Timbuk2 | 39.747552 | -104.999613 | 0.5 | Crowded | 1.0 | 0.555556 |
| **80202** | Alternative Healer | LoDo Wellness Center | 39.751730 | -104.999869 | 0.5 | Crowded | 1.0 | 0.555556 |
| **80202** | American Restaurant | Euclid Hall Bar & Kitchen | 39.747577 | -105.000243 | 0.5 | Crowded | 1.0 | 0.555556 |
| **80202** | American Restaurant | Freshcraft | 39.750012 | -104.999685 | 0.5 | Crowded | 1.0 | 0.555556 |
| **80202** | American Restaurant | Nickel | 39.746145 | -104.998202 | 0.5 | Crowded | 1.0 | 0.555556 |

. . .

# Results

​

The results of the model do/do not support my initial hypothesis that 80206 would be clustered similarly to 80211. However, it does highlight some other zipcodes which are modeled to be similar to 80211 based on the considered features. While we can see with the map that these zones often quickly extend outside the boundaries of Denver proper, which would likely disqualify them from our use case here, there are still a few of interest, most notably:

<ul><li>80205</li>

   <li>80207</li>

   <li>80220</li></ul>

# Results[¶](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb#Results)

The results of the model do/do not support my initial hypothesis that 80206 would be clustered similarly to 80211. However, it does highlight some other zipcodes which are modeled to be similar to 80211 based on the considered features. While we can see with the map that these zones often quickly extend outside the boundaries of Denver proper, which would likely disqualify them from our use case here, there are still a few of interest, most notably:

* 80205
* 80207
* 80220

# Discussion

​

While the model does not provide any single definitive "answer" for where an interested party should proceed to open a prospective restaurant, it is a useful tool for conducting their research more efficiently. The model highlights a subset of more eligible starting points than random chance, and also provides context for some of the features which might be influential or at least commonly associated with restaurant success.

[CloseExpandOpen in PagerClose](http://localhost:8888/notebooks/CapstoneProjectTheBattleOfNeighborhoods.ipynb)

References

1. “Colorado ZIP Code List”. Accessed 05/25/19 https://www.zipcodestogo.com/Colorado/
2. “FourSquare Developers API”. Accessed 05/25/19 https://developer.foursquare.com
3. “IRS SOI”. Accessed 05/25/19 <https://www.irs.gov/pub/irs-soi/16zp06co.xls>
4. “Colorado LightRail Stations”. Accessed 05/25/19 https://opendata.arcgis.com/datasets/17749050721d427399ab4e038028929d\_6.csv
5. “2010 Census Population by Zipcode (ZCTA).” Accessed 05/25/19 https://s3.amazonaws.com/SplitwiseBlogJB/2010+Census+Population+By+Zipcode+(ZCTA).csv
6. “Where can I find U.S. Census data with population per ZIP code? Other details such as age/gender breakdown would be helpful.” Accessed 05/25/19 https://www.quora.com/Where-can-I-find-U-S-Census-data-with-population-per-ZIP-code-Other-details-such-as-age-gender-breakdown-would-be-helpful
7. “Denver Statistical Neighborhoods.” Accessed 05/25/19 https://www.denvergov.org/content/denvergov/en/community-planning-and-development/planning-and-design/completed-plans/denver-statistical-neighborhoods.html
8. “More on k-means.” Accessed 05/25/19 https://www.coursera.org/learn/machine-learning-with-python/lecture/rLcgP/more-on-k-means
9. “LoHi Now Home to Over 75 Restaurants and Bars.” Accessed 05/25/19 https://www.westword.com/restaurants/lohi-now-home-to-over-75-restaurants-and-brs-11258151