ANALYZING NEIGHBORS IN COLORADO SKI RESORTS USING CLUSTERING

Abishek Coimbatore Sridhar
Colorado State University
Fort Collins, Colorado

INTRODUCTION

Colorado is one of the fast-growing states in US in terms of development and is expected to have the second fastest job growth over the next five years according to Forbes. It has the second highest level of college attainment, behind only Massachusetts[1]. Colorado is also called the "Ski Country USA", because of the endless skiing and snowboarding options throughout the Rocky Mountains that call the state home[2]. Resorts like Vail and Breckenridge offer world class skiing and snowboarding with luxurious lodging, dining, and nightlife in their base villages, while Arapahoe Basin and Loveland Ski Area offer hardcore skiing and a down to earth, local's vibe.

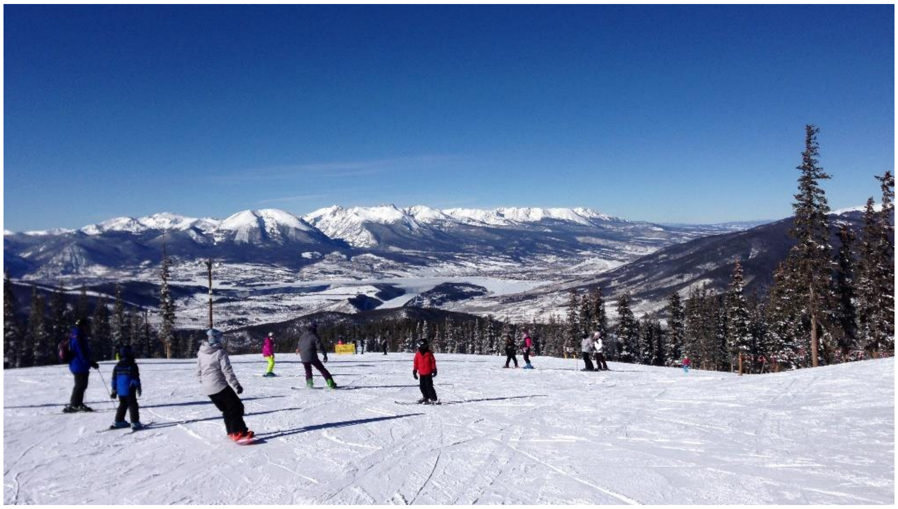

Fig 1: Ski Resort in Colorado

PROBLEM FORMULATION

There are 26 ski resorts in Colorado till today, the Capstone project provides a comprehensive comparison of the ski resorts with respect to the neighborhood of each ski resort. This project aims to provide clarity to customers coming from all over the world, to identify the ideal ski resort that suits the customer needs.

OBJECTIVE

The objective is to locate the ski resort that satisfy the customer needs in Colorado. It will also help the ski resort owners to select which of the neighborhood of Colorado ski resorts will be best choice to enlarge their operations. This would interest anyone who wants to visit or build their careers from the ski resorts in Colorado.

METHODOLOGY

This project is broken into several steps as followed:

- Extracting the Ski Resorts dataset down to only include area of Colorado from Wikipedia[3].
- Process the geographical data, to link each resort with a particular latitudes and longitudes in Colorado.
- Perform the final merging of the datasets, extracting key information from each dataset, like google reviews more than 1000 and nearest city.
- Based on the processed dataset, train a machine learning model using K-means Clustering model based upon their neighborhood in order to analyze the ideal locality near the ski resorts.

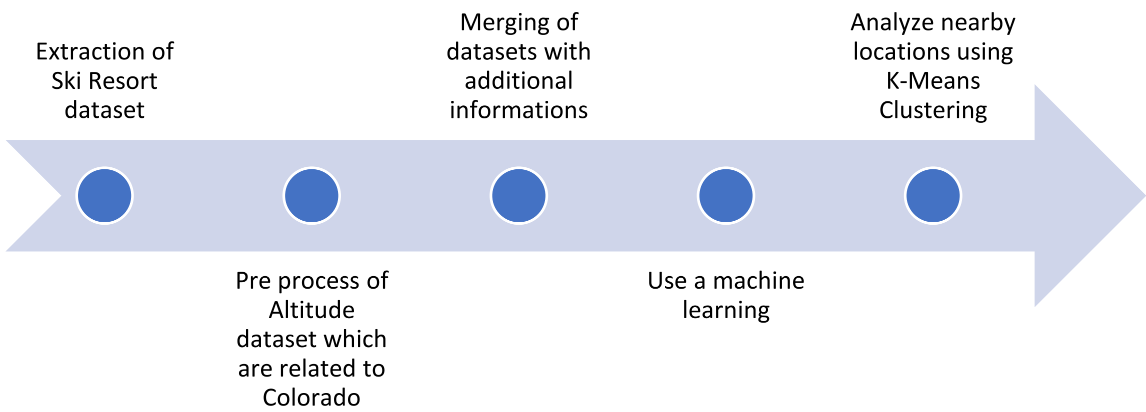

Fig 2: Methodology for this project

Title	***Source of dataset***
Ski Resort in USA	Wikipedia[3]
Latitude and Longitude of various cities in USA	OpenDataSoft[4]

Table 1: Source of dataset for this project

Data Extraction

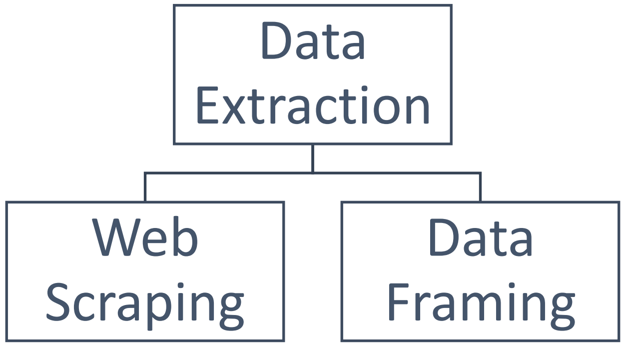

Fig 3: Classification of Data Extraction

The two process that involves in extraction of data are Web Scraping and Data Framing. In order to obtain information regarding Ski Resorts, we have to do web scraping from Wikipedia. Web scraping is a process of extracting a data from websites. Web scraping is also known as web harvesting or web data extraction[5]. Web scraping includes fetching of data from the website and extracting it for our use. Fetching is basically downloading the browser where web crawling can be utilized for later purposes. After fetching of data, extraction of data are done where contents are being parsed according to our needs and copy it in excel spreadsheets.

For web scraping in python, it is necessary to know requests and Beautiful Soup libraries which are the most powerful tools for web scraping. The requests library is useful for HTTP requests in python which returns responsive objects including content, encoding status and so on. Beautiful Soup is another python library for dragging the data from HTML and XML files which helps us to parse any information from those files[6].

First step for scrape a web page is to download the page. For downloading those web pages, we have used the requests library. The requests library will create a request to a web server which lets us download the HTML contents.

Installing Packages

In [2]:
```
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import requests
import geocoder
```

Fig 4: Installation of packages

For downloading the page, requests.get method is used in order to access the data from it [5]. Below, we have displayed the command for utilizing the data from Wikipedia for Ski Resorts in North America.

```
r = requests.get("https://en.wikipedia.org/wiki/Comparison_of_North_American_ski_resorts")
```

Fig 5: Command request.get from Wikipedia

After downloading the XML document, BeautifulSoup library is used to parse the document. Import the library and then create an instance of BeautifulSoup for parsing.

```
soup = BeautifulSoup(r.content, 'xml')
```

Fig 6: BeautifulSoup Library

For searching the data through the XML file, the nested tags can be used with the help of soup.select method. Below command shows stripping is done for elements with header of "th" and in "td" with "tr".

```
True)[0].strip() for head in soup.find_all("th")]
True)[0].strip() for td in tr.find_all("td")]for tr in table.find_all("tr")]for table in soup.select('table.wikitable.sortable')]
) for row in datas if len(row) == 12]for datas in data]
t1,columns=header)
```

Fig 7: Listing the elements from XML file

Second process is data framing where data is aligned in the form of rows and columns. Here we are dropping those rows which are not in Colorado. Fig 8 shows data framing and Fig 9 shows its elements

DataFrame cleaning

```
In [4]: #drop rows where 'Colorado is Not State/Province'
for index, value in enumerate(postal_df['State/province']!='Colorado'):
    if value:
        postal_df.drop([index], axis=0, inplace=True)
```

Fig 8: DataFrame Process

	Resort name and website	Nearest city	State/province	Peak elevation (ft)	Base elevation (ft)	Vertical drop (ft)	Skiable acreage	Total trails	Total lifts	Avg annual snowfall (in)	Adult weekend	Date statistics updated
0	Cooper	Leadville	Colorado	11,700	10,500	1,200	470	60	5	260	$70	September 19, 2019
1	Telluride Ski Resort	Telluride	Colorado	13,150	8,725	4,425	2,000	147	18	309	$149	December 11, 2019
2	Arapahoe Basin	Keystone	Colorado	13,050	10,520	2,530	1,428	147	9	350	$109	November 5, 2019
3	Loveland Ski Area	Georgetown	Colorado	13,010	10,800	2,210	1,800	94	11	422	$89	November 7, 2019
4	Keystone Resort	Keystone	Colorado	12,408	9,280	3,128	3,148	131	20	235	$169	December 11, 2019
5	Breckenridge Ski Resort	Breckenridge	Colorado	12,998	9,600	3,398	2,908	187	34	353	$189	December 11, 2019
6	Wolf Creek Ski Area	Pagosa Springs	Colorado	11,904	10,300	1,604	1,600	77	7	465	$76	December 11, 2019
7	Eldora Mountain Resort	Nederland	Colorado	10,800	9,200	1,600	680	53	12	300	$129	December 11, 2019
8	Steamboat Springs	Steamboat Springs	Colorado	10,568	6,900	3,668	2,956	165	16	349	$155	December 11, 2019
9	Copper Mountain (Colorado)	Frisco	Colorado	12,313	9,712	2,601	2,450	126	22	282	$190	December 11, 2019

Fig 9: List of elements

Preprocessing of Data

When processing the Ski Resort data, we first linked each resort with a zip code with their latitude and longitude point. We found the distance between each ski resort location and their associated zip code with the closest zip code centroid. Along with that, we omitted those values which are duplicate latitude and longitude as outliers in our final dataset. We added those latitudes and longitudes which are neighborhood to these Ski resorts.

Adding Latitude & Longitude for each of the Neighborhood.

```
In [116]: geocsv_data = pd.read_csv(r'C:\\Users\\sridh\\OneDrive\\Desktop\\us-zip-code-latitude-and-longitude.csv')
          geocsv_data
```

Out[116]:

	Zip	City	Latitude	Longitude
0	81148	Romeo	37.171583	-105.986480
1	81423	Norwood	38.082231	-108.258600
2	81505	Grand Junction	39.112514	-108.605060
3	81624	Collbran	39.220166	-107.934140
4	80402	Golden	39.522014	-105.223945
5	80235	Denver	39.648328	-105.084310
6	80620	Evans	40.378019	-104.703570
7	81001	Pueblo	38.282872	-104.570290
8	80202	Denver	39.751586	-104.996990
9	80454	Indian Hills	39.631254	-105.260710
10	81422	Naturita	38.296229	-108.680790

Fig 10: Latitude and Longitude of neighboring cities with zip codes

A merge of both the dataset is done using inner join and removal of outliers in order to get an accurate value of the neighborhood is taken in to account.

```
In [33]: denver_neighborhoods =pd.merge(denverPostalCodes, geocsv_data, how ='inner', on ='City')
         denver_neighborhoods

In [34]: denver_neighborhoods.drop([6,8,10,12,13,15,17,22,24,26,28,30,32,34,35,37,38,39,40,42,43,45,49,51,52,53,54,56,58,59], axis=0, inp
         denver_neighborhoods.reset_index(drop=True, inplace=True)
         #denver_neighborhoods.to_csv('Toronto.TASK_II_df.csv',index=False)
```

Fig 11: Merging of two datasets

Data Clustering

After gathering the data, it will be collected into a DataFrame and then use Folium package for visualizing the neighborhoods and their emerging clusters.

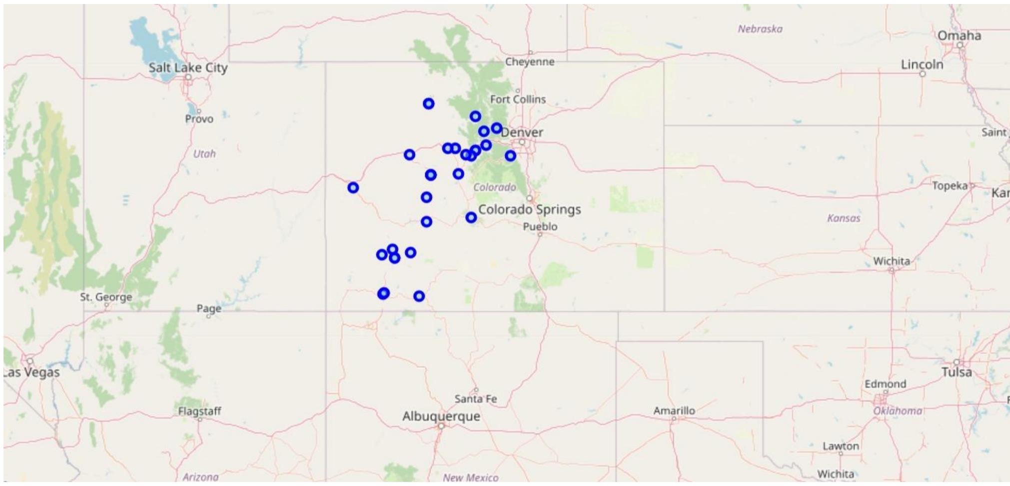

Fig 12: Colorado Map with its neighbors

Next, we have utilized the FourSquare API to retrieve nearby locations and search for important venues around the location. For allowing a user request, it is necessarily specifying Client Key's client ID and Client Secret in the request URL[7].

```
In [46]: #Utilizing the Foursquare API to explore and segment neighborhoods

CLIENT_ID = 'XSFGZXSRTDMNHRMHDRICCROK0RWGBZL2UJDUZIFTB5UJS3SR' # your Foursquare ID
CLIENT_SECRET = 'YKUKD1UBZA4KHNK0ARPJRUNKWYBHEVVYEVAWDINHZXOEUXYI' # your Foursquare Secret
VERSION = '20200604'
LIMIT = 30
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)

Your credentails:
CLIENT_ID: XSFGZXSRTDMNHRMHDRICCROK0RWGBZL2UJDUZIFTB5UJS3SR
CLIENT_SECRET:YKUKD1UBZA4KHNK0ARPJRUNKWYBHEVVYEVAWDINHZXOEUXYI
```

Fig 13: Defining the Foursquare API

Simplify the above map and cluster only the neighborhoods in Denver. So, let's reduce the original dataframe and create a new dataframe of the Denver data.

```
In [35]: address = 'Denver, Colorado USA'

geolocator = Nominatim(user_agent='My-IBMNotebook')
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Denver Colorado are {}, {}.'.format(latitude, longitude))

The geograpical coordinate of Denver Colorado are 39.7392364, -104.9848623.
```

Fig 14: Clustering of data to the neighbors of Denver

Later, we have listed the number of people rated with the ratings which are received from google reviews. Later, we have scrutinized the data by analyzing the number of ratings to be more than 1000 since less of reviewers with more ratings would not make it a reliable solution.

```
In [44]: denver_data = denver_neighborhoods[denver_neighborhoods['No.of people rated'].str.contains(',')].reset_index(drop=True)
         print(denver_data.shape)
         denver_data

(16, 17)
```

Fig 15: Scrutinization of data according to number of people rated

, 17)

Resort name and website	City	State/province	Peak elevation (ft)	Base elevation (ft)	Vertical drop (ft)	Skiable acreage	Total trails	Total lifts	Avg annual snowfall (in)	Adult weekend	Date statistics updated	Zip	Latitude	Longitude	No.of people rated	Ratings
Cooper	Leadville	Colorado	11,700	10,500	1,200	470	60	5	260.0	$70	19-Sep-19	80461	39.231776	-106.313990	1,549	4.6
Telluride Ski Resort	Telluride	Colorado	13,150	8,725	4,425	2,000	147	18	309.0	$149	11-Dec-19	81435	37.932874	-107.888740	1,901	4.8
Arapahoe Basin	Keystone	Colorado	13,050	10,520	2,530	1,428	147	9	350.0	$109	05-Nov-19	80435	39.604233	-105.948111	3,547	4.7
Keystone Resort	Keystone	Colorado	12,408	9,280	3,128	3,148	131	20	235.0	$169	11-Dec-19	80435	39.604233	-105.948111	2,802	4.6
Loveland Ski Area	Georgetown	Colorado	13,010	10,800	2,210	1,800	94	11	422.0	$89	07-Nov-19	80444	39.694915	-105.725800	3,053	4.7

Fig 16: Resultant data with number of people rated to be more than 1000

For analyzing the neighborhood, we applied the strategy of one hot encoding[8] to all the venues. So ,the number of columns becomes 48.

Analyze Each Neighborhood

```
In [52]: # one hot encoding
         denver_onehot = pd.get_dummies(denver_venues[['Venue Category']], prefix="", prefix_sep="")

         # add neighborhood column back to dataframe
         denver_onehot['Neighbourhood'] = denver_venues['Neighbourhood']

         # move neighborhood column to the first column
         fixed_columns = [denver_onehot.columns[-1]] + list(denver_onehot.columns[:-1])
         denver_onehot = denver_onehot[fixed_columns]

         denver_onehot.head()
```

Fig 17: One hot Encoding

Grouping the rows by neighborhood by taking the mean of frequency of number of occurrences that occurred in each category.

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

```
In [54]: denver_grouped = denver_onehot.groupby('Neighbourhood').mean().reset_index()
         denver_grouped
```

Fig 18: Grouping of rows

Out[59]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Avon	Mexican Restaurant	Grocery Store	Pizza Place	Liquor Store	Shipping Store	Furniture / Home Store	Electronics Store	Coffee Shop	Sandwich Place	Sushi Restaurant
1	Durango	Locksmith	Hotel	Coffee Shop	Video Store	Deli / Bodega	Indian Restaurant	Hot Dog Joint	Grocery Store	Golf Course	Furniture / Home Store
2	Georgetown	Train Station	Marijuana Dispensary	Hotel	Bar	Coffee Shop	Deli / Bodega	Indian Restaurant	Hot Dog Joint	Grocery Store	Golf Course
3	Keystone	Ski Area	Hotel	Coffee Shop	Sporting Goods Shop	Spa	Golf Course	Ski Lodge	Skating Rink	Dessert Shop	Hot Dog Joint
4	Leadville	Video Store	Construction & Landscaping	Indian Restaurant	Hotel	Hot Dog Joint	Grocery Store	Golf Course	Furniture / Home Store	Electronics Store	Dessert Shop
5	Nederland	Coffee Shop	American Restaurant	Indian Restaurant	Italian Restaurant	Jewelry Store	Marijuana Dispensary	Pizza Place	Plaza	Deli / Bodega	Restaurant
6	Vail	Ski Area	American Restaurant	Athletics & Sports	BBQ Joint	Hotel	Ski Chairlift	Construction & Landscaping	Hot Dog Joint	Grocery Store	Golf Course
7	Winter Park	Hotel	American Restaurant	Brewery	Indian Restaurant	Mexican Restaurant	New American Restaurant	Park	Coffee Shop	Chinese Restaurant	Pub

Fig 19: List of common venues

Let's put that into a pandas dataframe

In [56]:
```
num_top_venues = 5
for neigh in toronto_grouped['Neighbourhood']:
    print("----"+neigh+"----")
    temp = toronto_grouped[toronto_grouped['Neighbourhood'] == neigh].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----Avon----
                venue  freq
0  Mexican Restaurant  0.19
1         Pizza Place  0.12
2       Grocery Store  0.12
3         Coffee Shop  0.06
4      Sandwich Place  0.06
```

Fig 20: Venues with more frequency

Next, we have to cluster all the neighborhoods into different clusters which will allow us to identify which neighborhoods have higher concentration of venues while with a smaller number of venues. Using the highest number of venues in different neighborhoods will help us find out which neighborhoods are most common to visit near the Ski Resorts.

Clustering Neighborhoods

Run k-means to cluster the neighborhood into 6 clusters.

In [66]:
```
# set number of clusters
kclusters = 6

denver_grouped_clustering = denver_grouped.drop('Neighbourhood', 1)
# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=1).fit(denver_grouped_clustering)

# check cluster labels generated for each row in the dataframe
print(kmeans.labels_[0:10])
print(len(kmeans.labels_))
```

```
[3 2 5 0 1 3 4 3]
8
```

Fig 21: Clustering Neighborhoods

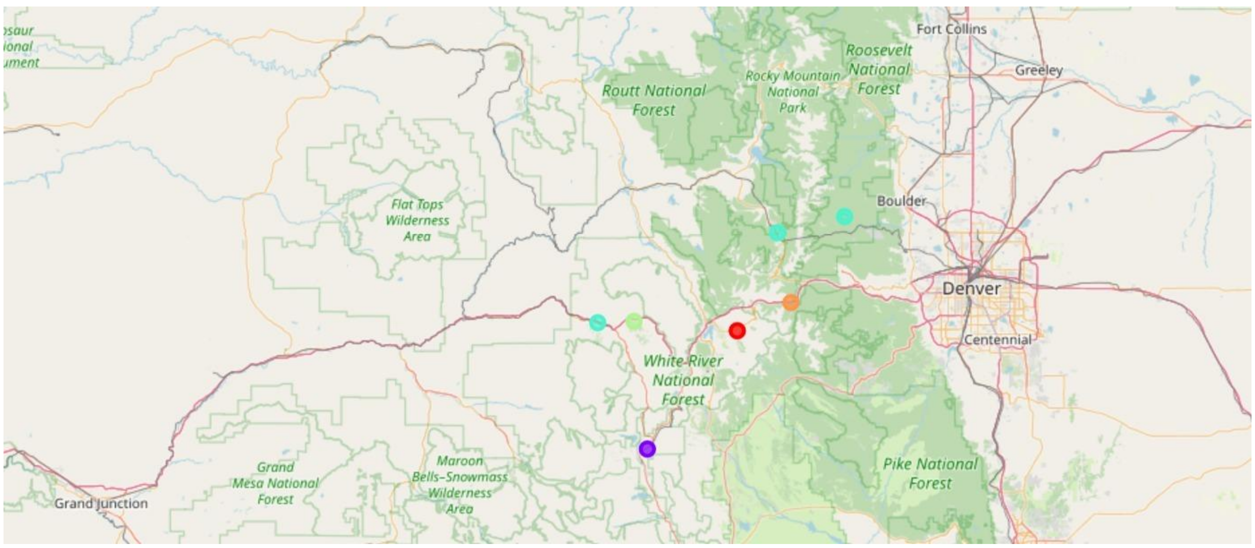

Fig 22: Visualizing the resulting clusters

RESULTS

The results from the above map shows us the categorization of clusters based on the similarities between top venues of different ski resorts. They are:

- Cluster 0 which has neighborhood with zero number of venues.(Red marker)
- Cluster 1 which has neighborhood of at least one venue. (Orange marker)
- Cluster 2 which has neighborhood of exactly two venues. (Violet marker)
- Cluster 3 which has neighborhood of more than two venues.(Green marker)

There are three places in cluster 3, and one place in rest of the clusters.

CONCLUSION

In this project , we utilized Colorado state and its neighborhoods based on the type of venues in the different neighborhoods. With the help of K-means clustering, the neighborhoods are categorized according to top 10 most common venues. Analyzing the clustering results obtained would help the skiers to have an idea on which place would be reliable to visit after their adventure. Future work for this project would be elaborating it with larger number of ski resorts across the world.

REFERENCES

1. Colorado. (n.d.). Retrieved July 24, 2020, from https://www.forbes.com/places/co/
2. Links to Area Attractions, Resources, News. (2020, June 21). Retrieved July 24, 2020, from https://rogersvacationrentals.com/insiders-guide-dillon-co/insiders-guide-dillon-co-links-to-area-attractions-resources-news/
3. List of ski areas and resorts in the United States. (2020, March 14). Retrieved July 24, 2020, from https://en.wikipedia.org/wiki/List_of_ski_areas_and_resorts_in_the_United_States
4. US Zip Code Latitude and Longitude. (2018, February 09). Retrieved July 24, 2020, from https://public.opendatasoft.com/explore/dataset/us-zip-code-latitude-and-longitude/table/?refine.state=CO
5. Mitchell, R. (2018). *Web scraping with Python: Collecting more data from the modern web*. " O'Reilly Media, Inc.".
6. Nair, V. G. (2014). *Getting Started with Beautiful Soup*. Packt Publishing Ltd.
7. Li, Y., Steiner, M., Wang, L., Zhang, Z. L., & Bao, J. (2013, April). Exploring venue popularity in foursquare. In *2013 Proceedings IEEE INFOCOM* (pp. 3357-3362). IEEE.
8. Raschka, S. (2015). *Python machine learning*. Packt publishing ltd.