Capstone Project – Technological and Innovation Center Location Recommendation

I. Introduction

A. Background

For any business, being in the right location is a key ingredient to its success. Choosing the wrong location may have a direct impact to access to customers, transportations and so on and consequently play a significant role in the company's profit and overall success. This remains true for expansion into new centers, branches, divisions, as the location plays a huge role in attracting and retaining the best employees, many of whom take it into consideration in order to optimize work-life balance. All this explain the importance of a location strategy which is a plan aiming at obtaining the optimal location for a company by identifying the company needs and objectives, and searching for locations with offerings that are compatible with these needs and objectives. Good location decisions can significantly boost a company's long-term performance. Poor ones can cost millions in lost talent, productivity and capital. Very often, companies have to consider, among others, the below points when making this strategic decision:

- The size and diversity of the community, the cultural fit and quality of life to attract and retain talent
- The logistics to gauge the accessibility of the location
- The availability of a strong university system for future labor pool
- The stability and business-friendliness of the communities from a legal and regulatory standpoint
- The capital and operating expenses
- The competition

B. Business Problem

The objective of this capstone project is to analyze and recommend cities in the US to a company whose goal is to open a second Technological and Innovation Center as part of its expansion plan that should offer them the similar facilities as the current center located in Seattle, Washington. Using data science and machine learning techniques we aim at answering the following question: *Which US cities meet the company's requirements and would be best suited to house the new technological center?* The chosen city must meet the following requirements:

- Major city.
- Logistics, this includes:
  - Presence of at least one international airport.
  - Good mass transit system
- Cultural community fit, this includes:
  - Quality of life
  - Safety
  - Attractions/Recreational opportunities
  - Cost of living
  - Diversity
- Workforce/Labor pool, this includes:
  - An educated population

▪ A strong university system

We will exclude the laws and regulations, operating expenses and competition criteria from our analysis

C. Target Audience

We believe this project to be particularly useful to companies, entrepreneurs looking for the right location for their businesses and whose criteria fit the above listed. This might also be beneficial to cities officials who wish to make their cities more competitive to attract new businesses.

II. Methodology

In this project we will direct our efforts on identifying which cities meet the needs expressed and would be best suited to house the second technological center.

- ✓ In the first step we will collect all the required data, and perform cleansing and all other pre-processing steps to ensure that all input are at the city level and aggregated accordingly. To minimize the number of calls on the Foursquare API, recreational activities data would be added later on to the last cities
- ✓ In the second step we will consolidate all input dataset into one for analysis purpose. To ensure that we only analyze the applicable cities we will limit our analysis to those that at least meet the requirements like large cities, international airports, labor pool and so on...
- ✓ In the third step, we will use **features engineering** to identify the most relevant features to use in our model
- ✓ In the fourth and final step, with Seattle Washington as our benchmark, we will cluster the cities to identify which ones have similar characteristics as our benchmark using **k-means clustering**. We will also, using the correlation coefficient, rank our recommended cities

III. Analysis

A. First Step - Data Acquisition and Cleansing

The goal of this first step was to acquire and pre-process the data to facilitate its analysis in the second phase. The objective was to have each dataset at the city- state level since this was the subject of our project.

1) Data Sources

Based on definition of our problem, the factors that will influence our decision are:

- the size of the city in terms of population

- the number of airport present in or around the city
- the number and type of attractions/recreational opportunities in or around the city
- the number of crimes
- the score or ranking of the city mass transit system, quality of life, cost of living, education system and business friendliness

The following data sources will be needed to extract/generate the required information:

- Population: http://worldpopulationreview.com/us-cities/
- Airports: https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_the_United_States
- Education data: https://wallethub.com/edu/most-and-least-educated-cities/6656/
- Crimes: https://ucr.fbi.gov/crime-in-the-u.s/2017/crime-in-the-u.s.-2017/additional-data-collections/federal-crime-data
- Mass transit system: https://alltransit.cnt.org/rankings/
- Universities data: https://ope.ed.gov/dapip/#/download-data-files and https://www.mastersportal.com/ranking-country/82/united-states.html
- Diversity: https://wallethub.com/edu/most-diverse-cities/12690/
- Cost of living: https://www.numbeo.com/cost-of-living/region_rankings.jsp?title=2018&region=021
- Quality of life: https://www.numbeo.com/quality-of-life/rankings.jsp
- Business Friendliness: https://wallethub.com/edu/best-cities-to-start-a-business/2281/
- Attractions: Foursquare API

2) Data Cleansing

After downloading the data the next step was to pre-process it. For ease of understanding, the data was cleansed individually (by type) prior its consolidation into a single dataset.

In addition to renaming and dropping unnecessary columns as applicable, the following transformations were applied to the data:

Universities data:

- Limit the dataset to only accredited and ranked universities
- Extract from the addresses the City and State and create a City_State column that would be used as the join column later on when creating the master dataframe
- Group the data at the city level with the creation of the following statistics number of universities in the city, and the least and most ranked university in the city
- Remove all duplicates records since the data was already at the city level

Population data:

- Create a City_State column that would be used as the join column later on when creating the master dataframe by join the population data to the state name to code dataset to pull the state code.

- Replace all abbreviations (for example St. instead of Saint) and special character (for example –) in the city names

<u>Airports data:</u>

- Join the City and State columns to create our City_State joining condition
- Group the data at the city level with the creation of the following statistics number of airports in the city, average number of passengers, total number of passengers and the least and most ranked airport in the city
- Remove all duplicates records since the data was already at the city level

<u>Quality of life data:</u>

- Limit our data to cities in the United States
- Extract from the location column the City_State joining condition

<u>Cost of living data:</u>

- Limit our data to cities in the United States
- Extract from the location column the City_State joining condition

<u>Crimes data:</u>

- Create a City_State column that would be used as the join column later on when creating the master dataframe by join the population data to the state name to code dataset to pull the state code.
- Replace all abbreviations (for example St. instead of Saint) and special character (for example –) in the city names
- Change the datatype of number of crimes columns from Object to Float
- Replace all missing Arson crimes with 0 for the cities

<u>Mass transit data:</u>

- Replace all abbreviations (for example St. instead of Saint) and special character (for example –) in the city names

<u>Business friendliness data</u>

- Replace all abbreviations (for example St. instead of Saint) and special character (for example –) in the city names

<u>Diversity data:</u>

- Replace all abbreviations (for example St. instead of Saint) and special character (for example –) in the city names

<u>Education data</u>

- Replace all abbreviations (for example St. instead of Saint) and special character (for example –) in the city names

B. Second Step - Data Consolidation

The above listed cleansed datasets were later on combined into one by the City_State joining condition. Given the differences in the number of records in each dataset, there were a lot of missing values for multiple features. I decided to start including some of our requirements to ensure that the analysis was only done on relevant cities. The following thresholds were applied to our dataset:

- Population should be at least equal to 250,000 habitants
- Number of airports should be at least equal to 1
- Number of universities should be at least equal to 1

The resulting dataset had 41 contender cities. And fewer null values than our original dataset

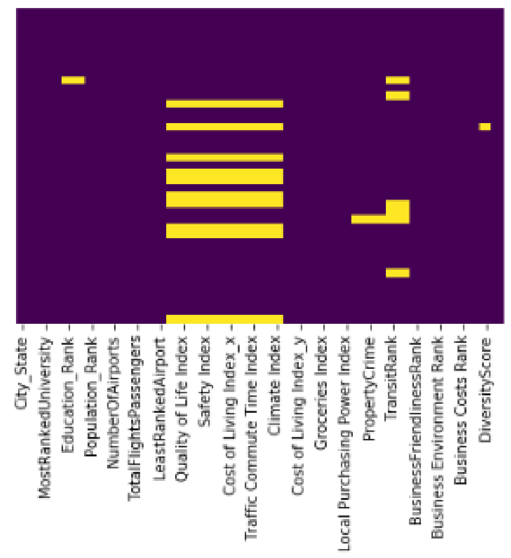


Figure 1: Missing values check

I plotted the correlation heatmap for the dataset and identified the presence of a lot of strong correlations among the cost of living and quality of life variables, as well as the crimes variables. This led me to believe that some of those variables could be excluded from our model as they might provide the same type of information. I decided to apply **features engineering** concepts to identify the most relevant variables. But before that we will incorporate the recreational activities data.

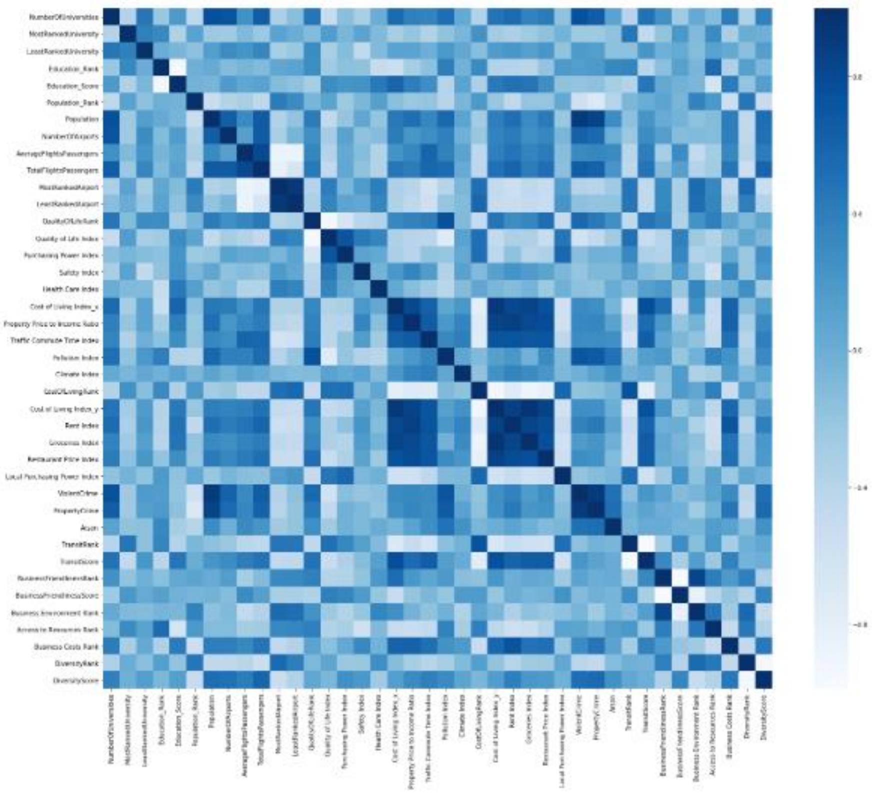


Figure 2: Correlation heatmap

Recreational activities data:

To source the recreational activities data we used the Foursquare API. We set out limits to 100 venues within a 500miles radius of the city coordinates.

Using our 41 cities, I pulled the names (format City Name, State Code) into a dictionary. I created an empty dataframe and using the *Nominatim* library from the *geopy geocoders* module pulled each city longitude and latitude information. That data was then used to pull venues names, categories, latitude, longitude, city name and state name from Foursquare using the API and saved them in our dataframe.

The info function showed the presence of null values in the City name column and looking at those records, I used google to populate the city name for those attractions.

Special character (.) was also found in the state code for the District of Columbia and replace with nothing to have DC instead of D.C.

To allow me to join this dataset to our existing master dataframe, I:

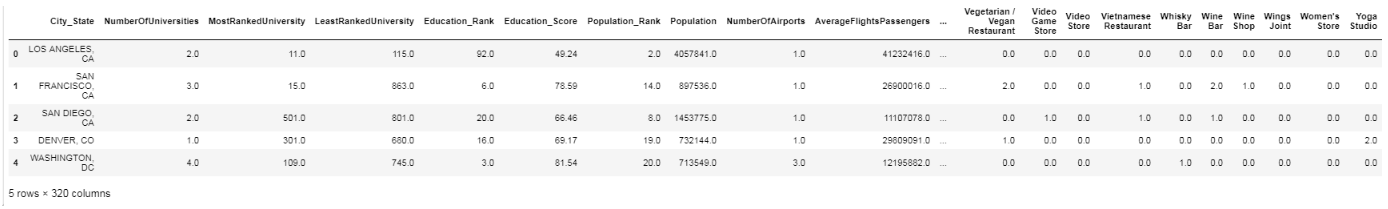
- Created the City_State joining condition by concatenating the City name and State Code from the venues dataframe

- Grouped the number of venues by category and city to have the information at the city level
- Used the pivot functionality to transform our dataframe to be at the city level instead of at the venue level as the example below shows

Before pivot and aggregation			After pivot and aggregation		
Venue Name	**Category Name**	**City_State**	**City_State**	**Park**	**Hotel**
ABC Park	Park	Charlotte, NC	**Charlotte, NC**	1	2
Hilton	Hotel	Charlotte, NC			
Meridian	Hotel	Charlotte, NC			

Table 1: Grouping of venues at the city level

I then merged that data with our master dataframe. Resulting dataframe had 41 rows and 318 columns.



	City_State	NumberOfUniversities	MostRankedUniversity	LeastRankedUniversity	Education_Rank	Education_Score	Population_Rank	Population	NumberOfAirports	AverageFlightsPassengers	...
0	LOS ANGELES, CA	2.0	11.0	115.0	92.0	49.24	2.0	4057841.0	1.0	41232416.0	...
1	SAN FRANCISCO, CA	3.0	15.0	863.0	6.0	78.59	14.0	897536.0	1.0	26900016.0	...
2	SAN DIEGO, CA	2.0	501.0	801.0	20.0	66.46	8.0	1453775.0	1.0	11107078.0	...
3	DENVER, CO	1.0	301.0	680.0	16.0	69.17	19.0	732144.0	1.0	29809091.0	...
4	WASHINGTON, DC	4.0	109.0	745.0	3.0	81.54	20.0	713549.0	3.0	12195882.0	...

	Vegetarian / Vegan Restaurant	Video Game Store	Video Store	Vietnamese Restaurant	Whisky Bar	Wine Bar	Wine Shop	Wings Joint	Women's Store	Yoga Studio
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2.0	0.0	0.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0
2	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

5 rows × 320 columns

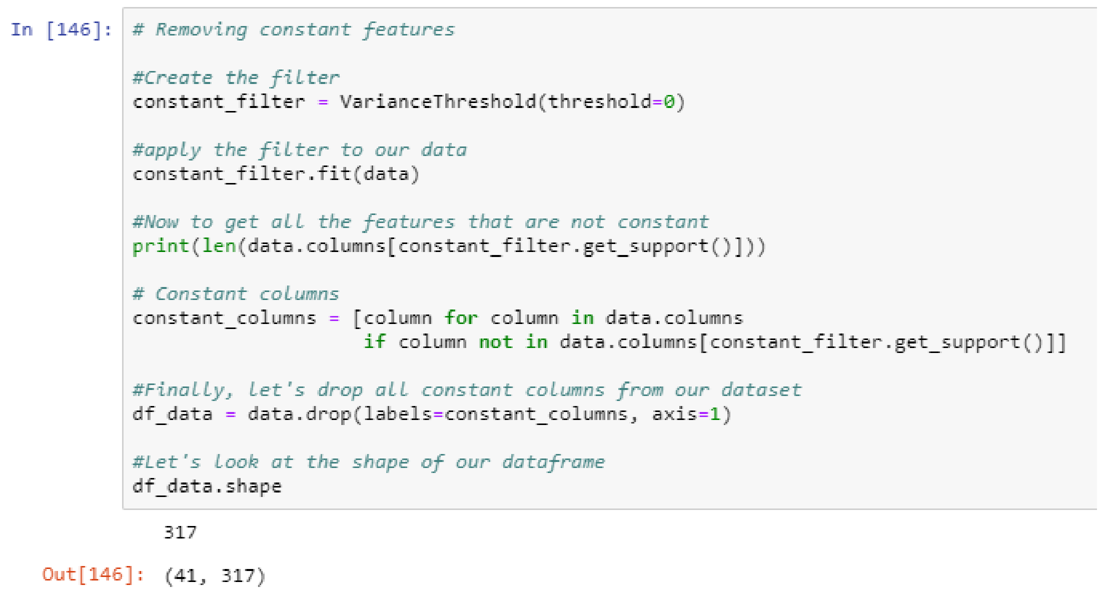
Figure 3: Master dataframe

C. Third Step - Features Selection

After data cleaning, there were 41 cities and 318 features in the data. Upon examining the correlation heatmap, it was clear that there was some redundancy in the features. For example the cost of living and quality of life variables had strong correlations among them, as well as the crimes variables. I therefore decided to apply feature engineering techniques to select only the most relevant ones.

The first step was to fill the null values; I decided to replace them with 0 to note that they were not available.

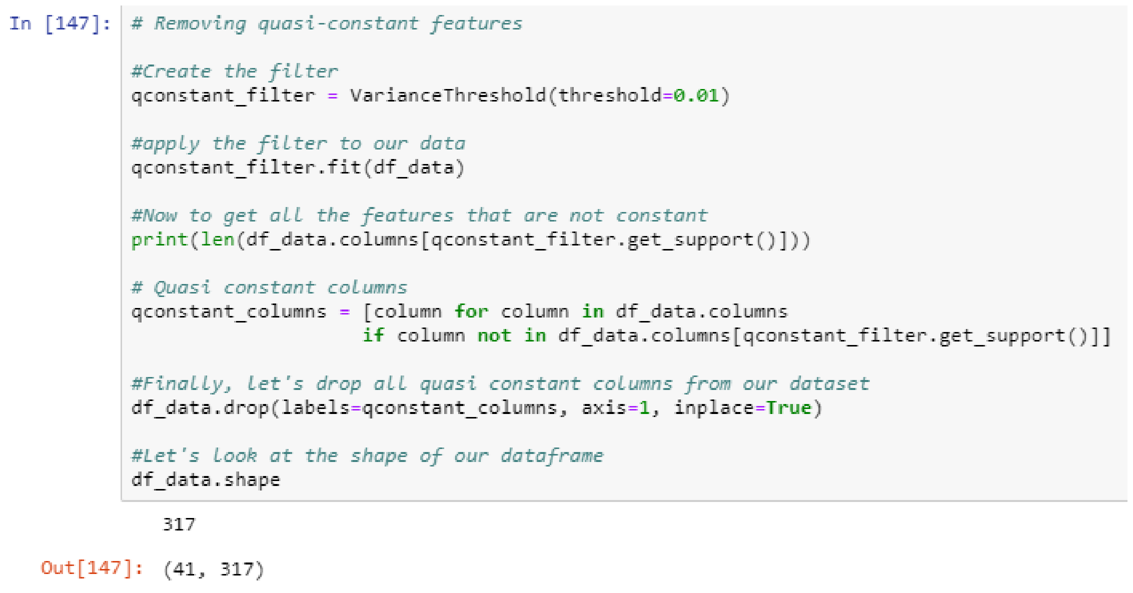
The next step was to remove constant features. Constant features are defined as features with values with zero variance. To do that, I used the *VariableThreshold* library from the *Sklearn feature_selection* module and set my threshold to 0. This step removed 1 feature from the dataset.



```
In [146]: # Removing constant features

          #Create the filter
          constant_filter = VarianceThreshold(threshold=0)

          #apply the filter to our data
          constant_filter.fit(data)

          #Now to get all the features that are not constant
          print(len(data.columns[constant_filter.get_support()]))

          # Constant columns
          constant_columns = [column for column in data.columns
                              if column not in data.columns[constant_filter.get_support()]]

          #Finally, let's drop all constant columns from our dataset
          df_data = data.drop(labels=constant_columns, axis=1)

          #Let's look at the shape of our dataframe
          df_data.shape

          317

Out[146]: (41, 317)
```

Figure 4: Constant features removal

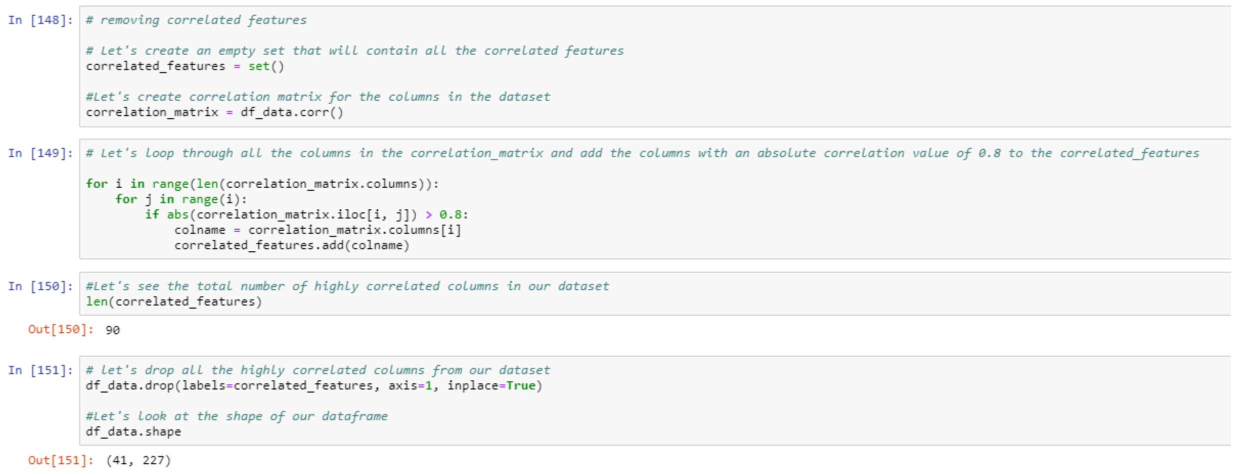
The next step was to remove quasi-constant features. Quasi-constant features as defined as features that are almost constant. To do that, I used the *VariableThreshold* library from the *Sklearn feature_selection* module and set my threshold to 0.01. This step did not remove any feature from the dataset.



```
In [147]: # Removing quasi-constant features

          #Create the filter
          qconstant_filter = VarianceThreshold(threshold=0.01)

          #apply the filter to our data
          qconstant_filter.fit(df_data)

          #Now to get all the features that are not constant
          print(len(df_data.columns[qconstant_filter.get_support()]))

          # Quasi constant columns
          qconstant_columns = [column for column in df_data.columns
                               if column not in df_data.columns[qconstant_filter.get_support()]]

          #Finally, let's drop all quasi constant columns from our dataset
          df_data.drop(labels=qconstant_columns, axis=1, inplace=True)

          #Let's look at the shape of our dataframe
          df_data.shape

          317

Out[147]: (41, 317)
```

Figure 5: Quasi-constant features removal

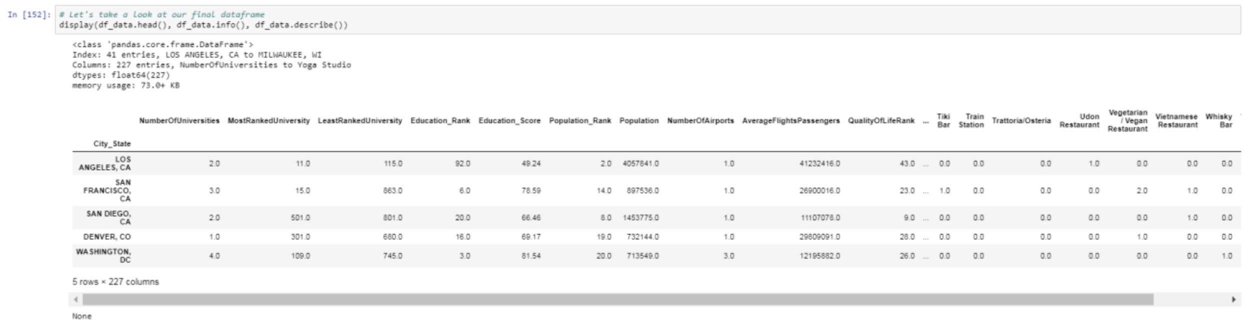
The final step was to remove highly correlated features. To do that, I used the correlation matrix and removed features with a correlation coefficient greater to absolute value of 0.8. This step removed 90 features from our dataset.



```
In [148]: # removing correlated features

          # Let's create an empty set that will contain all the correlated features
          correlated_features = set()

          #Let's create correlation matrix for the columns in the dataset
          correlation_matrix = df_data.corr()

In [149]: # Let's loop through all the columns in the correlation_matrix and add the columns with an absolute correlation value of 0.8 to the correlated_features

          for i in range(len(correlation_matrix.columns)):
              for j in range(i):
                  if abs(correlation_matrix.iloc[i, j]) > 0.8:
                      colname = correlation_matrix.columns[i]
                      correlated_features.add(colname)

In [150]: #Let's see the total number of highly correlated columns in our dataset
          len(correlated_features)

Out[150]: 90

In [151]: # Let's drop all the highly correlated columns from our dataset
          df_data.drop(labels=correlated_features, axis=1, inplace=True)

          #Let's look at the shape of our dataframe
          df_data.shape

Out[151]: (41, 227)
```

Figure 6: Highly correlated features removal

At the end of our feature engineering process 91 features had been removed from our dataset, leaving me with 227 features.



```
In [152]: # Let's take a look at our final dataframe
          display(df_data.head(), df_data.info(), df_data.describe())

<class 'pandas.core.frame.DataFrame'>
Index: 41 entries, LOS ANGELES, CA to MILWAUKEE, WI
Columns: 227 entries, NumberOfUniversities to Yoga Studio
dtypes: float64(227)
memory usage: 73.0+ KB
```

City_State	NumberOfUniversities	MostRankedUniversity	LeastRankedUniversity	Education_Rank	Education_Score	Population_Rank	Population	NumberOfAirports	AverageFlightsPassengers	QualityOfLifeRank	...	Tiki Bar	Train Station	Trattoria/Osteria	Udon Restaurant	Vegetarian / Vegan Restaurant	Vietnamese Restaurant	Whisky Bar
LOS ANGELES, CA	2.0	11.0	115.0	92.0	49.24	2.0	4057841.0	1.0	41232416.0	43.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0
SAN FRANCISCO, CA	3.0	15.0	863.0	6.0	78.59	14.0	897536.0	1.0	26900016.0	23.0	...	1.0	0.0	0.0	0.0	2.0	1.0	0.0
SAN DIEGO, CA	2.0	501.0	801.0	20.0	66.46	8.0	1453775.0	1.0	11107078.0	9.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0
DENVER, CO	1.0	301.0	680.0	16.0	69.17	19.0	732144.0	1.0	29809091.0	28.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0
WASHINGTON, DC	4.0	109.0	745.0	3.0	81.54	20.0	713549.0	3.0	12195882.0	26.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0

```
5 rows × 227 columns

None
```

Figure 7: Master dataframe after features selection process

D.Fourth Step – Cities Classification and Ranking

In the fourth and final step, with Seattle Washington as our benchmark, we will use **k-means clustering** to cluster the cities together in order to identify which ones have similar characteristics as our benchmark. We will also, using the **correlation coefficient**, rank our recommended cities by order of similarities.

1) Clustering

Using our selected features, I initially clustered the cities into 3 groups. Using the elbow method, I checked the optimal number of clusters for our dataset and confirmed that using k = 3 was the optimal choice for our dataset.

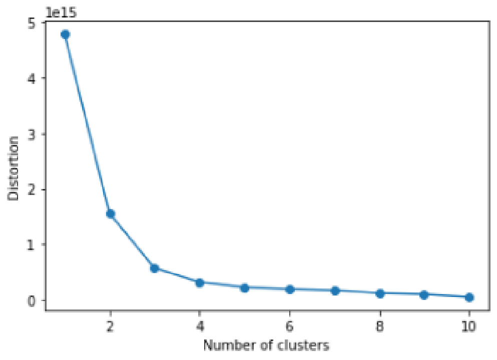
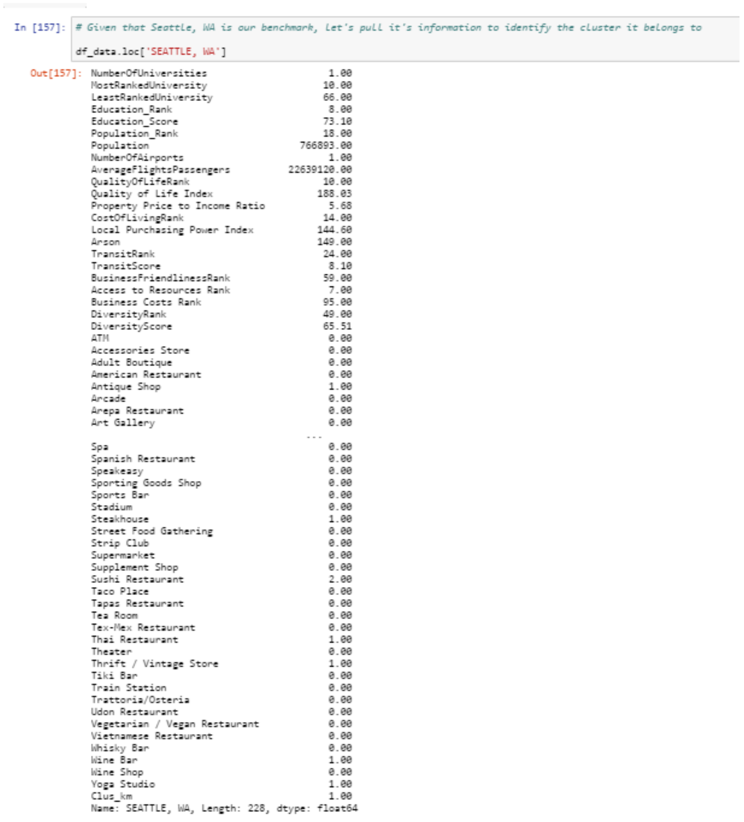


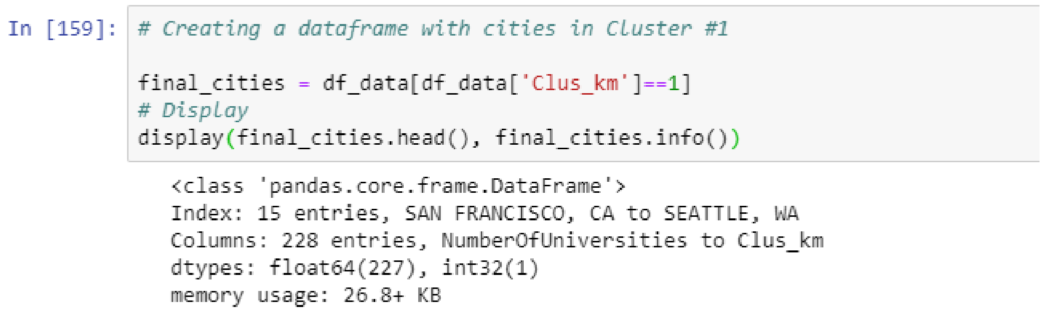
Figure 8: The Elbow method identifying the optimal K

Following the clustering of cities into each group, the labels were added in our dataset to identify the label assigned to our benchmark city, Seattle, WA. A check of our dataframe helped us identify that Seattle, WA was assigned to cluster label 1 along with 14 other cities.



```
In [157]: # Given that Seattle, WA is our benchmark, let's pull it's information to identify the cluster it belongs to

          df_data.loc['SEATTLE, WA']

Out[157]: NumberOfUniversities                    1.00
          MostRankedUniversity                   10.00
          LeastRankedUniversity                  66.00
          Education_Rank                          8.00
          Education_Score                        73.10
          Population_Rank                        18.00
          Population                         766893.00
          NumberOfAirports                        1.00
          AverageFlightsPassengers         22639120.00
          QualityOfLifeRank                      10.00
          Quality of Life Index                 188.03
          Property Price to Income Ratio          5.68
          CostOfLivingRank                       14.00
          Local Purchasing Power Index          144.60
          Arson                                 149.00
          TransitRank                            24.00
          TransitScore                            8.10
          BusinessFriendlinessRank               59.00
          Access to Resources Rank                7.00
          Business Costs Rank                    95.00
          DiversityRank                          49.00
          DiversityScore                         65.51
          ATM                                     0.00
          Accessories Store                       0.00
          Adult Boutique                          0.00
          American Restaurant                     0.00
          Antique Shop                            1.00
          Arcade                                  0.00
          Arepa Restaurant                        0.00
          Art Gallery                             0.00
                                              ...
          Spa                                     0.00
          Spanish Restaurant                      0.00
          Speakeasy                               0.00
          Sporting Goods Shop                     0.00
          Sports Bar                              0.00
          Stadium                                 0.00
          Steakhouse                              1.00
          Street Food Gathering                   0.00
          Strip Club                              0.00
          Supermarket                             0.00
          Supplement Shop                         0.00
          Sushi Restaurant                        2.00
          Taco Place                              0.00
          Tapas Restaurant                        0.00
          Tea Room                                0.00
          Tex-Mex Restaurant                      0.00
          Thai Restaurant                         1.00
          Theater                                 0.00
          Thrift / Vintage Store                  1.00
          Tiki Bar                                0.00
          Train Station                           0.00
          Trattoria/Osteria                       0.00
          Udon Restaurant                         0.00
          Vegetarian / Vegan Restaurant           0.00
          Vietnamese Restaurant                   0.00
          Whisky Bar                              0.00
          Wine Bar                                1.00
          Wine Shop                               0.00
          Yoga Studio                             1.00
          Clus_km                                 1.00
          Name: SEATTLE, WA, Length: 228, dtype: float64
```

Figure 9: Seattle, WA features



```
In [159]: # Creating a dataframe with cities in Cluster #1

          final_cities = df_data[df_data['Clus_km']==1]
          # Display
          display(final_cities.head(), final_cities.info())

            <class 'pandas.core.frame.DataFrame'>
            Index: 15 entries, SAN FRANCISCO, CA to SEATTLE, WA
            Columns: 228 entries, NumberOfUniversities to Clus_km
            dtypes: float64(227), int32(1)
            memory usage: 26.8+ KB
```

Figure 10: Dataframe of cities labelled 1

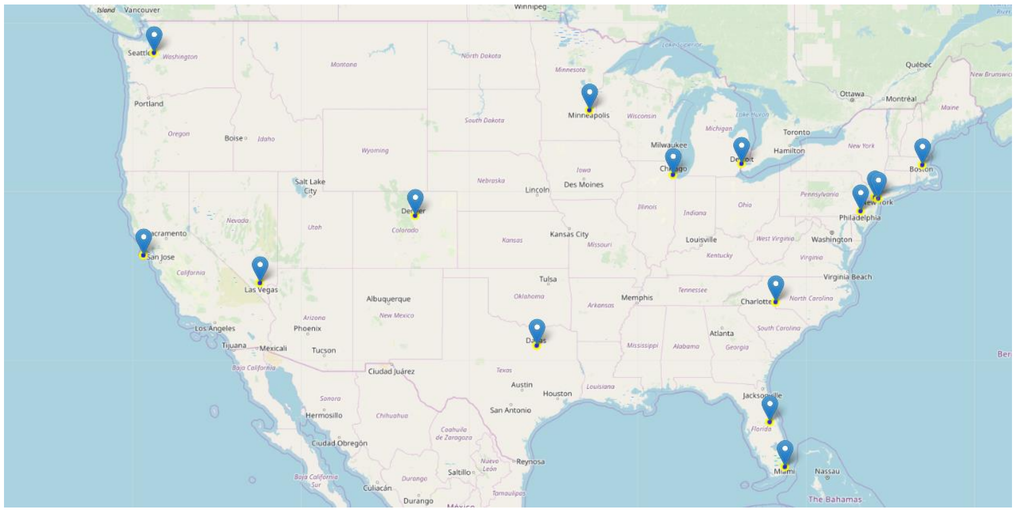
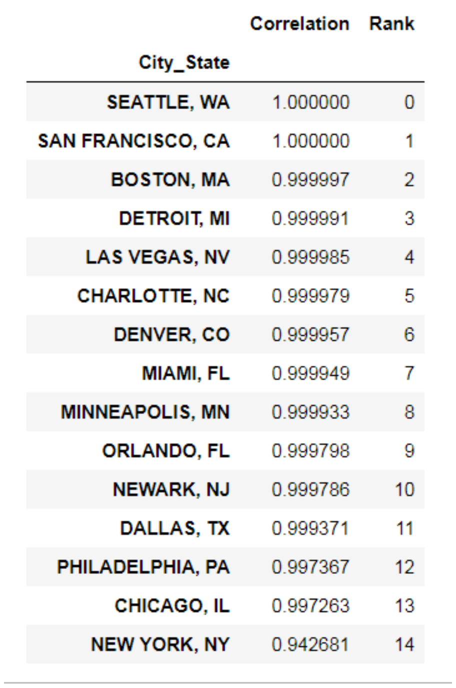


Figure 11: Map of Recommended cities

2) Ranking

The final step was to rank the final cities relative to their similarity to Seattle, WA. To do that I computed the correlation coefficient of the cities with Seattle, WA and ordered the results, saved in a dataframe, in descending order. The result is as follow:



City_State	Correlation	Rank
SEATTLE, WA	1.000000	0
SAN FRANCISCO, CA	1.000000	1
BOSTON, MA	0.999997	2
DETROIT, MI	0.999991	3
LAS VEGAS, NV	0.999985	4
CHARLOTTE, NC	0.999979	5
DENVER, CO	0.999957	6
MIAMI, FL	0.999949	7
MINNEAPOLIS, MN	0.999933	8
ORLANDO, FL	0.999798	9
NEWARK, NJ	0.999786	10
DALLAS, TX	0.999371	11
PHILADELPHIA, PA	0.997367	12
CHICAGO, IL	0.997263	13
NEW YORK, NY	0.942681	14

Table 2: Recommended Cities Ranking

3) Results

Our analysis shows that 13 cities could potentially answer the needs of the company among the approximately 4000 cities we started with.

The correlation results show that San Francisco, CA characteristics are extremely similar to Seattle's however the decision the company would have to make is to decide if they want to branch out on the East Cost of the country in which case Boston, MA would be the best candidate.

IV. Conclusion

The purpose of this project was to identify and recommend the best cities to house a second Technological Center that would have similar characteristics as Seattle, WA in order to aid stakeholders in narrowing down the search for optimal location. By using several sources that were in line with the company's requirements we have first identify 41 cities that would meet the basic needs of the company.

Clustering of those locations was then performed in order to further reduce the list to the cities with similar characteristics as Seattle, WA.

The final decision on optimal location will be made by stakeholders based on specific characteristics in every recommended city, taking into consideration additional factors like the business environment, incentives and competitors, real estate availability and etc...

V. Future directions

I think the model could use some improvements by capturing additional features for each city like the business environment, incentives and competitors, real estate availability, network coverage, weather, etc... More data, especially related to the quality of life, cost of living and mass transit could significantly help improve the model.