

|  |
| --- |
| Machine Learning |
| *Exploring the Taste of Chicago* |
| February 11  Coursera  Authored by: Utkarsh Kapoor |

# Introduction

Background

Chicago is the most popular city in the United States, home to the headquarters of the United Nations and an important center for international diplomacy. It just might be the most diverse city on the planet, as it is home to over 8.6 million people and over 800 languages. As quoted in an article - What Food Tells Us About Culture “Traditional cuisine is passed down from one generation to the next. It also operates as an expression of cultural identity. Immigrants bring the food of their countries with them wherever they go and cooking traditional food is a way of preserving their culture when they move to new places.”

Problem

Undoubtedly, Food Diversity is an important part of an ethnically diverse metropolis. The idea of this project is to categorically segment the neighborhoods of Chicago into major clusters and examine their cuisines. A desirable intention is to examine the neighborhood cluster's food habits and taste. Further examination might reveal if food has any relationship with the diversity of a neighborhood. This project will help to understand the diversity of a neighborhood by leveraging venue data from Foursquare’s ‘Places API’ and ‘k-means clustering’ unsupervised machine learning algorithm. Exploratory Data Analysis (EDA) will help to discover further about the culture and diversity of the neighborhood.

Stakeholders

This quantifiable analysis can be used to understand the distribution of different cultures and cuisines over ‘the most diverse city on the planet – Chicago’. Also, it can be utilized by a new food vendor who is willing to open his or her restaurant. Or by a government authority to examine and study their city's culture diversity better.

# Data

# To examine the above said, following data sources will be used:

# Chicago Dataset

# Link: <https://en.wikipedia.org/wiki/List_of_neighborhoods_in_Chicago>

# There are sometimes said to be more than 200 neighborhoods in Chicago, though few residents would agree on their names and boundaries. A city ordinance prescribing and mapping 178 neighborhoods is almost unknown and ignored even by municipal departments. Neighborhood names and identities have evolved over time due to real estate development and changing demographics. The City of Chicago is also divided into 77 community areas which were drawn by University of Chicago researchers in the late 1920s. Chicago's community areas are well-defined, generally contain multiple neighborhoods, and are less commonly used by city residents. More historical images of Chicago neighborhoods can be found in Explore Chicago Collections, a digital repository made available by Chicago Collections archives, libraries and other cultural institutions in the city. Here is an example of webpage table data:

# 

# Foursquare API

# Link: <https://developer.foursquare.com/docs>

# Foursquare API, a location data provider, will be used to make RESTful API calls to retrieve data about venues in different neighborhoods. This is the link to Foursquare Venue Category Hierarchy. Venues retrieved from all the neighborhoods are categorized broadly into ‘Arts & Entertainment’, ‘College &University’, ‘Event’, ‘Food’, ‘Nightlife Spot’, ‘Outdoors & Recreation’, etc. An extract of an API call is as follows:

# 

# Methodology

# Download and Explore Chicago Dataset

# Here we will consider the original Wikipedia page for creating our data set. We will use Beautiful soup to get the table and create a pandas data frame from there.

# 

# Now we will include the latitude and longitude for each Borough.

# 

# Further, ‘geopy’ library is used to get the latitude and longitude values of Chicago City. The curated dataframe is then used to visualize by creating a map of Chicago City with neighborhoods superimposed on top. The following depiction is a map generated using python ‘folium’ library.

# 

# RESTful API Calls to Foursquare

The Foursquare API is used to explore the neighborhoods and segment them. To access the API, ‘CLIENT\_ID’, ‘CLIENT\_SECRET’ and ‘VERSION’ is defined. There are many endpoints available on Foursquare for various GET requests. But, to explore the cuisines, it is required that all the venues extracted are from ‘Food’ category. Foursquare Venue Category Hierarchy is retrieved using the following code block:

# 

# The returned request is further analyzed:

# 

# Upon analysis, it is found that there are 10 major or parent categories of venues, under which all the other sub-categories are included. Following depiction shows the ‘Category ID’ and ‘Category Name’ retrieved from API:

# 

# As said earlier, the ‘FOOD’ category in the above depiction is the matter of interest. A function is created to return a dictionary with ‘Category ID’ & ‘Category Name’ of ‘Food’ & it's sub-categories.

# 

# This above function takes the parent ‘Category ID’ and returns the ‘Category Name’ and ‘Category ID’ of all the sub-categories.

To further understand the results of GET Request, the first neighborhood of the ‘Chicago City’ dataset is explored. The first neighborhood returned is ‘North Mayfair’ with Latitude 41.71 and Longitude - 87.69. Then, a GET request URL is created to search for Venue with ‘Category ID’ = '4d4b7105d754a06374d81259', which is the ‘Category ID’ for ‘Food’, and radius = 500 meters.



The returned request is then examined, which is as follows:



The category name of the venue 'Chi Tung Restaurant' is 'Food' which is returned here.

As, the aim is to segment the neighborhoods of Chicago City with respect to the ‘Food’ in its vicinity, it is further required to fetch this data from all the neighborhoods' venues.

To overcome the redundancy of the process followed above, a function ‘getNearbyFood’ is created. This functions loop through all the neighborhoods of Chicago City and creates an API request URL with radius = 500, LIMIT = 100. By limit, it is defined that maximum 100 nearby venues should be returned. Further, the GET request is made to Foursquare API and only relevant information for each nearby venue is extracted from it. The data is then appended to a python ‘list’. Lastly the python ‘list’ is unfolded or flattened to append it to data frame being returned by the function. It is inquisitive to know that Foursquare API returns all the sub-categories, if a top-level category is specified in the GET Request.

# 

# Pickle

# Pickle is a very important and easy-to-use library. It is used to serialize the information retrieved from GET requests, to make a persistent ‘.pkl’ file. This file can later be deserialized to retrieve an exact python object structure. This is a crucial step as it will counter any redundant requests to the Foursquare API, which is chargeable over the threshold limits.

# 

# The returned ‘dataframe’ is as follows:

# 

# As of now, two python ‘dataframe’ are created:

# 1) ‘neighborhoods’ which contains the Borough, Neighborhood, Latitude and Longitude details of the Chicago City’s neighborhood, and

# 2) ‘chi\_venues’ which is a merger between ‘neighborhoods’ dataframe and its ‘Food’ category venues searched with ‘Radius’ = 500 meters and ‘Limit’ = 100. Also, each venue has its own Latitude, Longitude and Category.

# Exploratory Data Analysis

# The merged dataframe ‘chi\_venues’ has all the required information. The size of this dataframe is determined, and it is found that there are total 3556 venues.

# Now, it is important to find out that how many unique categories can be curated from all the returned venues. There are 109 such categories, with most occurring venues as follows:

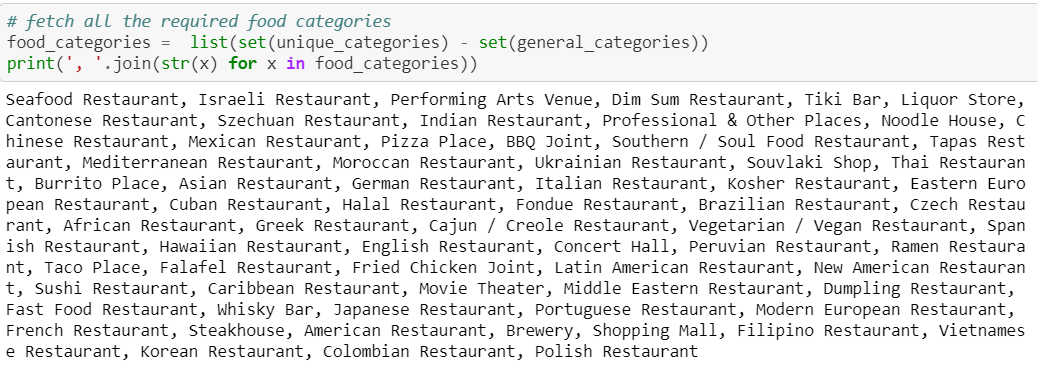
# 

# Data Cleaning

# It is crucial to understand that the point of interest in the project is to understand the cultural diversity of a neighborhood by clustering it categorically, using the venues’ categories. Thus, it is important to remove all the venues from the ‘dataframe’ which have generalized categories. Here, by generalized, it means that these categorized venues are common across different cultures and food habits. Example of categories of this type of venues are Coffee Shop, Cafe, etc. So, firstly all the unique categories are fed into a python ‘list’.

# 

Then, manually the categories are determined to be ‘general’ (as explained above). This data pre-preparation totally depends upon the ‘Data Analyst’ discretion and can be modified as required. Following are the categories listed as ‘general’:



# 

# A simple subtraction of two python ‘list’ i.e ‘unique\_categories’ and ‘general\_categories’ gives a ‘list’ of all the categories which are required for further analysis. Following image depicts the result of the above activity:

# 

# The python ‘list’ curated above, is used to remove all the venues with categories not in ‘food\_categories’, and the following dataframe is retrieved:

# 

# Again, the number of unique categories is examined, and it is found that there are only 68 of them, as compared to 108 earlier. That means, almost 40% of the data was a noise for the analysis. This essential step, data cleaning, helped to capture the data points of interest.

# Feature Engineering

# Now, each neighborhood is analyzed individually to understand the most common cuisine being served within its 500 meters of vicinity. The above process is taken forth by using ‘one hot encoding’ function of python ‘pandas’ library. One hot encoding converts the categorical variables (which are ‘Venue Category’) into a form that could be provided to ML algorithms to do a better job in prediction.

# 

# Upon converting the categorical variables, as shown above, ‘Neighborhood’ column is added back which results into the following:

# 

# Further, number of venues of each category in each neighborhood are counted.

# 

# The top 10 ‘Venue Categories’ can also be found by counting their occurrences. This analysis is depicted below which shows that ‘Mexican Restaurant’, ‘Fast food Restaurant’,

# Fried Chicken Restaurant’, ‘Pizza Restaurant’, and ‘Fast Food Restaurant’ are among the top 5.

# Data Visualization

# These top 10 categories are further plotted individually on bar graph using python ‘seaborn’ library. The following code block creates the graph of top 10 neighborhoods for a category.

# 

# 

# Next, the rows of the neighborhood are grouped together and the frequency of occurrence of each category is calculated by taking the mean.

# 

# As the limit is set to be 100, there will be many venues being returned by the Foursquare API. But a neighborhood food habit can be defined by the top 5 venues in its vicinity. Following ‘for’ loop creates a dataframe to record the abovementioned data points:

# 

# Further, the above created dataframe is fed with the top 5 most common venues categories in the respective neighborhood.

# 

# Machine Learning

# ‘k-means’ is an unsupervised machine learning algorithm which creates clusters of data points aggregated together because of certain similarities. This algorithm will be used to

# count neighborhoods for each cluster label for variable cluster size. To implement this algorithm, it is very important to determine the optimal number of clusters (i.e. k). There

# are 2 most popular methods for the same, namely ‘The Elbow Method’ and ‘The Silhouette Method’.

# *The Elbow Method*

# The Elbow Method calculates the sum of squared distances of samples to their closest cluster center for different values of ‘k’. The optimal number of clusters is the value after

# which there is no significant decrease in the sum of squared distances. Following is an implementation of this method (with varying number of clusters from 1 to 49):

# 

# 

# Sometimes, Elbow method does not give the required result, which happened in this case. As, there is a gradual decrease in the sum of squared distances, optimal number of

# clusters cannot be determined. To counter this, another method can be implemented, as discussed below.

# *The Silhouette Method*

# As quoted in Wikipedia – “The Silhouette Method measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).” Following is an

# implementation of this method. As it requires minimum 2 clusters to define dissimilarity number of clusters (i.e. ‘k’) will vary from 2 to 49:

# 

# 

# k-Means

# Following code block runs the k-Means algorithm with number of clusters = 8 and prints the counts of neighborhoods assigned to different clusters:

# 

# Further the cluster labels curated are added to the dataframe to get the desired results of segmenting the neighborhood based upon the most common venues in its vicinity:

# 

# Now, ‘neighborhoods\_venues\_sorted’ is merged with ‘nyc\_data’ to add the Borough, Latitude and Longitude for each neighborhood.

# 

# Again, the Chicago City’s neighborhoods are visualized by using the code block as shown, which utilizes the python ‘folium’ library.

# 

# Following map is generated which shows the desired segmentation of the Chicago’s neighborhoods:

# 

# Results

# Cluster 0

# 

# Following are the results of the Cluster – 0 analysis:

# 

# Cluster 1

# 

# Cluster 2

# 

# Cluster 3

# 

# Cluster 4

# 

# Cluster 5

# 

# Cluster 6

# 

# Cluster 7

# 

# Discussion

# To understand the clusters, three analysis were done, namely:

# 1. Count of ‘Borough’

# 2. Count of ‘1st Most Common Venue’

# 3. Count of ‘2nd Most Common Venue’

# The above information speaks a lot about the ground reality of clustering based on the similarity metrics between the neighborhoods.

# Tabulating the results of the k-Mean unsupervised machine learning algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| Count of Occurrences within the Cluster | | | |
| Cluster | 1st Most Common Venue | 2nd Most Common Venue | Borough |
| 0 | American Restaurant | Italian Restaurant | Ashburn, Belmont Cragin, Morgan Park, Bridgeport, Brighton Park |
| 1 | Mexican Restaurant | Fast Food Restaurant | Hyde Park, Irving Park, Lincoln Square, West Elsdon, West Ridge |
| 2 | Pizza Place | American Restaurant | Uptown, Auburn Gresham, Lakeview |
| 3 | Fast Food Restaurant | American Restaurant | West Englewood, Lake View, Garfield Ridge, West Garfield Park, East Garfield Park |
| 4 | Pizza Place | Fast Food Restaurant | Albany Park, Armour Square, East Side, Grand Boulevard, North Lawndale |
| 5 | Fried Chicken Joint | Fast Food Restaurant | Chicago Lawn, South Chicago, Forest Glen |
| 6 | American Restaurant | Fast Food Restaurant | North Park, Washington Park |
| 7 | Mexican Restaurant | Pizza Place | Austin, South Lawndale, Humboldt Park |

# Fast food, who does not like it. And it is obvious from the analysis that Fast Food Restaurant is the most common venue across all the clusters or neighborhoods.

# So, as Fast food is a ready-to-go place for Chicago City, it is kept aside to rename the clusters.

# Following could be the name of the clusters segmented and curated by k-Means unsupervised machine learning algorithm:

# Cluster 0 - American Restaurant

# Cluster 1 - Mexican Restaurant

# Cluster 2 - Pizza Place

# Cluster 3 - Fast Food Restaurant

# Cluster 4 - Pizza Place

# Cluster 5 - Fried Chicken Joint

# Cluster 6 - American Restaurant

# Cluster 7 - Mexican Restaurant

# Conclusion

# On application of Clustering Algorithm, k-Means or others, to a multi-dimensional dataset, a very inquisitive results can be curated which helps to understand and visualize the data. The neighborhoods of Chicago City were very briefly segmented into eight clusters and upon analysis it was possible to rename them basis upon the categories of venues in and around that neighborhood. Along with the American cuisine, Italian and Chinese are very dominant in Chicago City and so is the diversity statistics.

# The results of this project can be improved and made more inquisitive by using a current Chicago City’s dataset along with API platforms which are more interested in Food Venues (like Yelp, etc.) The scope of this project can be expanded further to understand the dynamics of each neighborhood and suggest a new vendor a profitable location to open his or her food place. Also, a government authority can utilize it to examine and study their city's culture diversity better.