Recommendation System Using Credit Card Transactions

Capstone 2

Fariha Baloch
Springboard - Data Science Career Track

Problem Statement

Create a recommendation system using credit card transactions records. The data contains transactions from multiple people and contains GPS records. The aim is to provide the customer following

a. The recommendation system should recommend similar merchants to the customer based on similarities between the merchants
b. Based on the location of a merchant, other merchants in the vicinity are recommended to a user

Data Mining

- The data for this project came from FinGoal.
- FinGoal analyzes consumer spending to give radically insightful recommendations built on the experience of other savvy consumers (source: FinGoal.com).
- The data was anonymized and converted to json format by the company. The file size was 48.2MB.
- Used pd.read_json to read the file into a data frame. The file had 29 columns and 90086 rows of transactions.

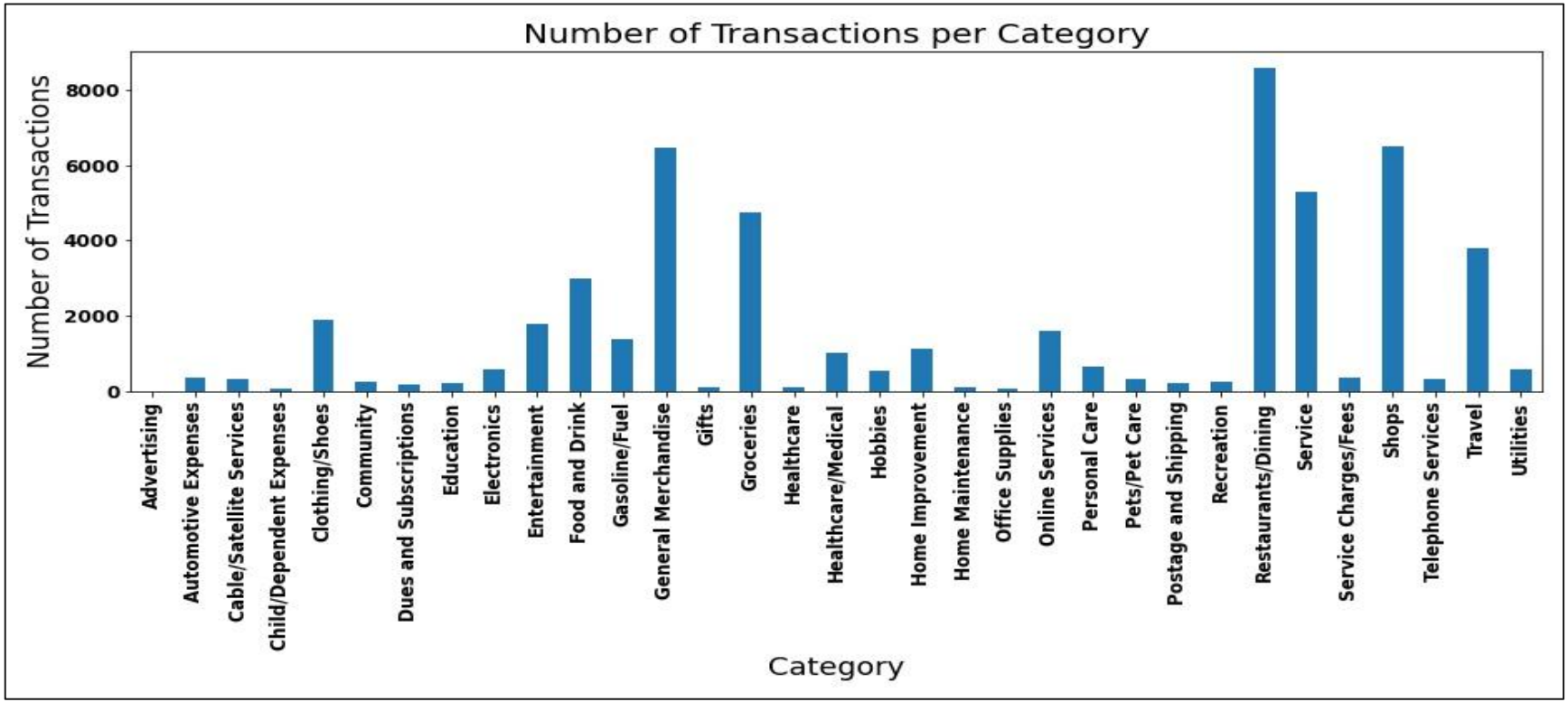
Data Wrangling

- Dropped columns with no values and columns that were not important to the recommendation system
- Used columns like zip_code and city to fill NaNs of each other
- Multiple uid's were missing accountid. Therefore assigned random unique numbers to their accountid in order to get rid of the NaNs in accountid columns
- Category column had values that were related to expenditures by a customer, for example, PayChecks/Salary. All such values were deleted from the data by deleting the rows with those specific column values
- In the ammountnum column 75% of dollar amount spent by consumers was less than $52. A few of the consumer purchases were > $6000 and were considered outliers therefore those rows (44) were deleted

Exploratory Data Analysis - EDA

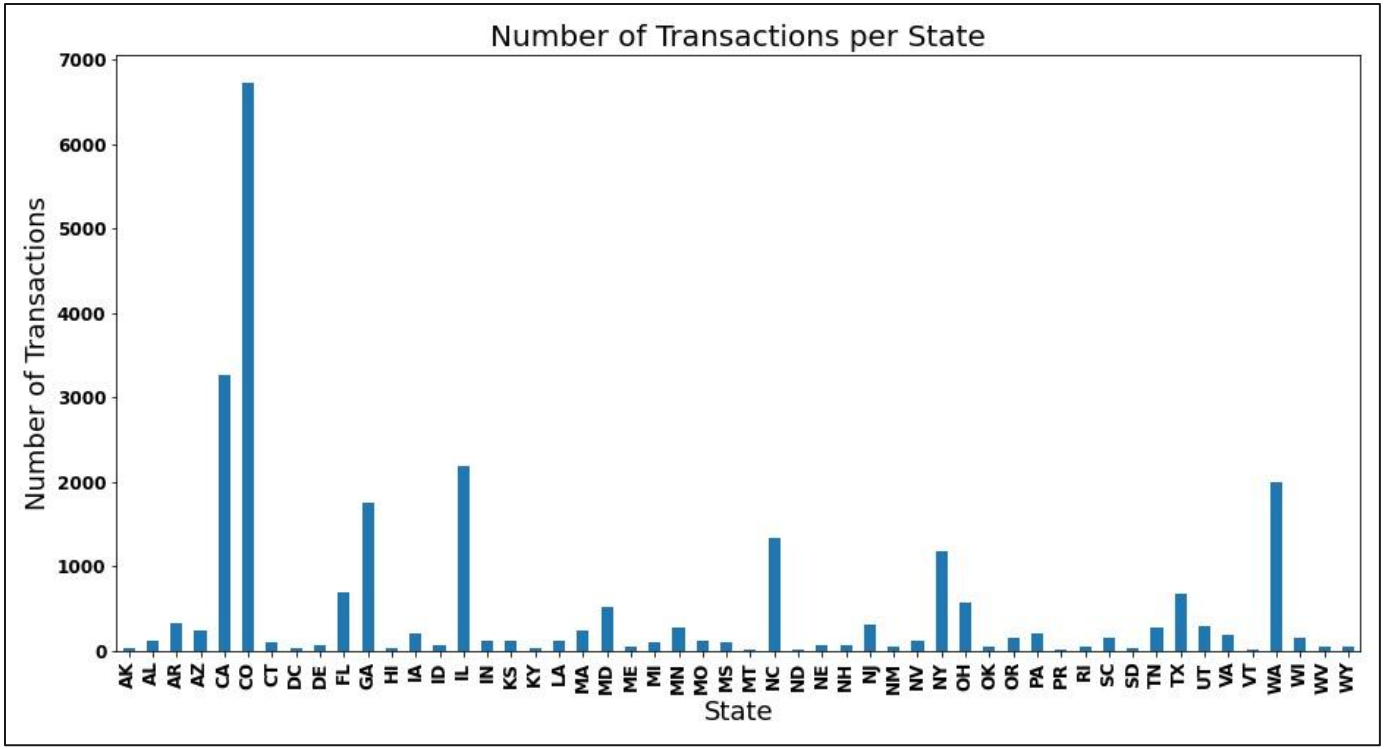
Exploratory Data Analysis

- To see how the transaction were divided into categories and states a few bar charts were created
- Several date/time graphs were plotted to understand how the data changed from month to month



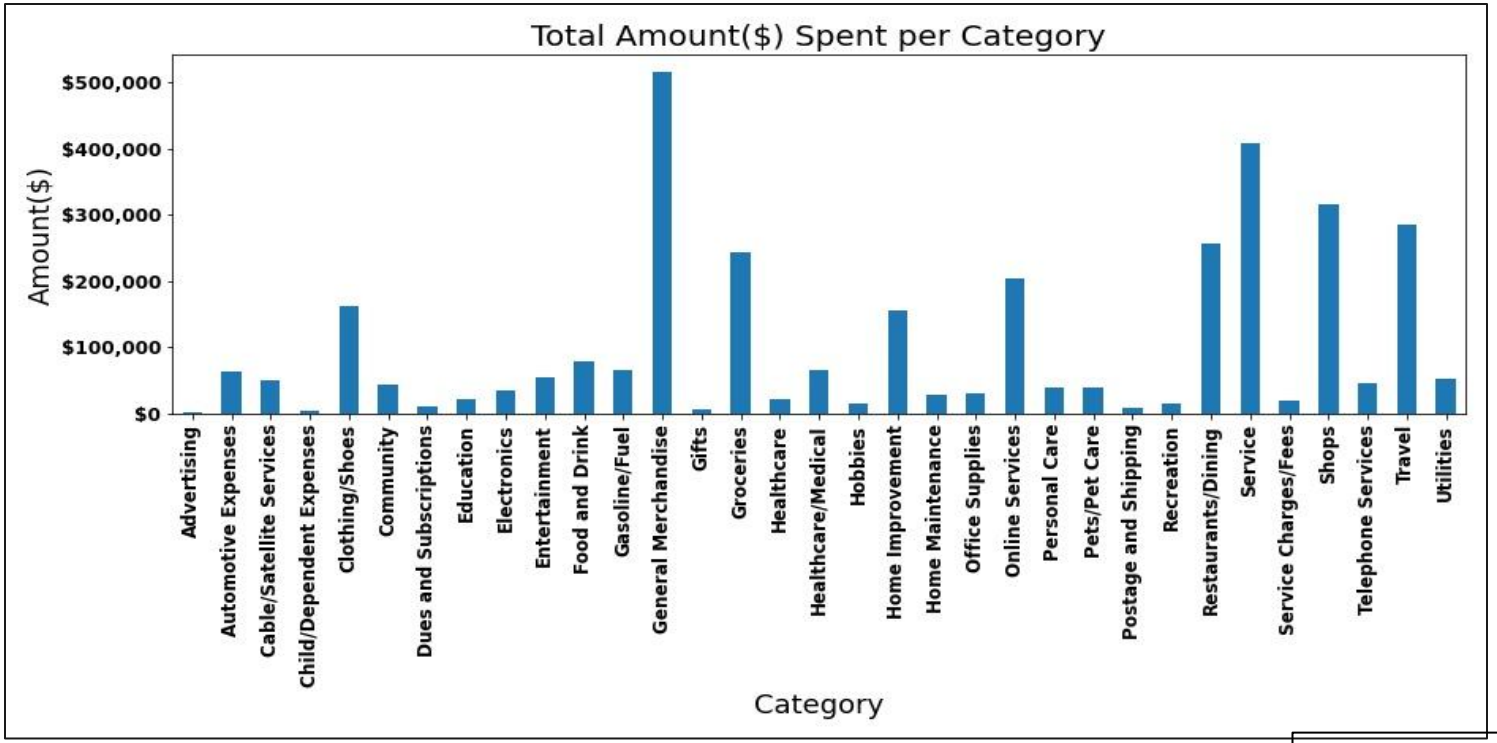
Restaurants/Dining have most transactions (>8k)

General Merchandise, Shops, Service and Groceries follow next



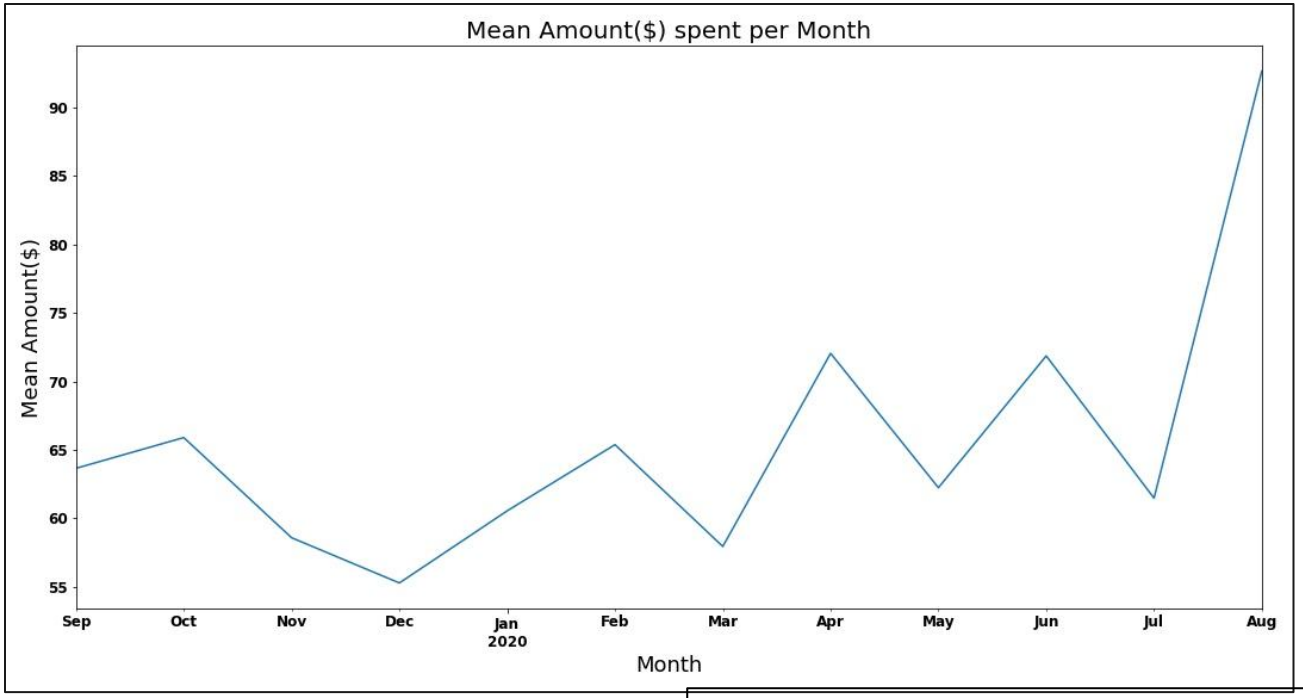
CO has the most transactions(~7k)

Majority of the states have less than 500 transactions



General Merchandise brought in close to $500k

Service, Shops, Travel,Restaurants/Dining and Groceries are some categories where customer spent between $250k - $400k



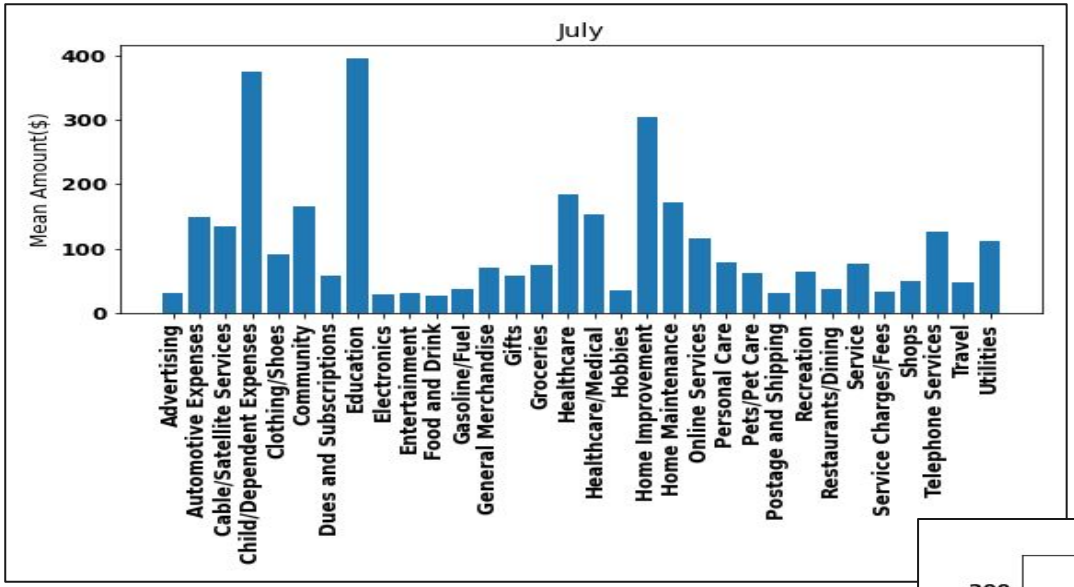
DateTime Analysis

Clear spikes in the data can be noticed in different months

In March a huge dip occurs - Due to Corona sales went down

In April the data shows a high mean amount, however most of the transactions are in “Office Supply” category. Corona did affect this month as well but not in all categories

In August there is a huge mean amount increment from July - Users spent in all categories



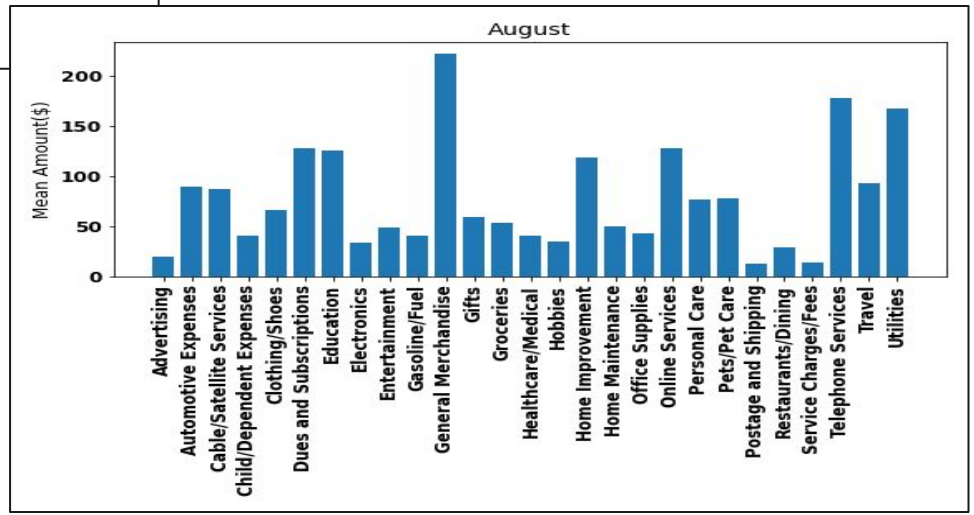
Total Transactions July - 19666
Total Transactions Aug -2090

Total Transactions Sum July - $120, 8925
Total Transactions Sum Aug - $193. 644

General Merchandise - mean amount increased in August compared to July (~$100 to~$200) - reason might be the start of school year

Child Dev Expenses decreased in Aug (~$400 to ~$50) - reason might be the start of school year

Overall there was increase in spending in Aug from amount($) perspective. However the number of transaction were lesser than July ….Therefore a spike in Aug Mean Amount



Goal 1: Collaborative Filtering

Based on a user’s preference, recommend similar merchants

Recommendation System V1

- Goal 1:
  - The recommendation system should recommend similar merchants to the customer based on similarities between the merchants

Recommendation System - Collaborative Filtering

Collaborative Filtering: The process of collaborative filtering for a recommendation system relies on information (ratings, count etc) collected by users in the system.

For example, to recommend a movie to a user who has not watched it before, the recommendation system collects ratings from the users that have watched it and are similar to the specific user and recommends the movie (or not) to that specific user.

For this project, the value between rows (uid) and columns(merchant) is the total number of times the user visited a merchant. Not a rating given to a merchant.

Some more Data Wrangling

- Imported df1 from EDA.ipynb
- Went through Simple_description column and extracted the names of the merchant as much as possible.
    - Deleted word like ‘pending’, ‘visa’ and all other forms on these before the merchant names,
    - This helped with consolidating same merchants together
- There are 33 categories and there are 13 categories with > 900 transactions
- Only those categories were used to build the recommendation system
- Each category has several number of merchants therefore a data frame with top 15 merchants is created to train the models

Steps

- A merchant with which the similarity is supposed to match is selected
- Create filter(s) apart from the similarity score on which the final recommendation should be posted
    - Count per merchant : Popularity of the merchant among other users i.e.e count of the times users went to this merchant
    - Mean amount per merchant: Average amount spent by users on this merchant
- A sparse matrix is created with rows as uid and columns as merchants. The values in cells are the number of times the specific merchant has been visited by a user
- A series is created that show only shows popularity (number of visits)of the merchant in question with all the users

Models

To evaluate similarity between merchants two methods were used

1. **Pearson Correlation**

   Using Pandas method ‘data frame. corrwith()’ pairwise correlation between rows of two dataframes is calculated. This produced a series of numbers between -1 and 1, showing correlation between all the merchants with the specific merchant. A number close to 1 or -1 shows high similarity.

2. **Cosine Similarity Score**

   Using sklearn’s function ‘cosine_similarity’, compute the similarity score between the merchants. The result is nxn sparse matrix, where n is total number of merchants in refined category data. A number closer to 1 shows high similarity and vice versa for the number closer to 0

   *A depiction of how merchant similarity score is calculated is in Backup Slides

Final Step

Using the defined threshold for acceptable similarity score and for the defined filters ,show the results that match the criteria

The filter looks like this:

Output_list=
(**merchant_similarity_score** >= Threshold_Sim) &
(**Num_Times_Merch_Was_Visited** > Threshold_Num_Visit) &
(**Mean_Amt_Per_Merch** > Threshold_Num_MeanAmt) &
(**merchant_recommended** != merchant_in_question)

Goal 2: Collaborative Filtering

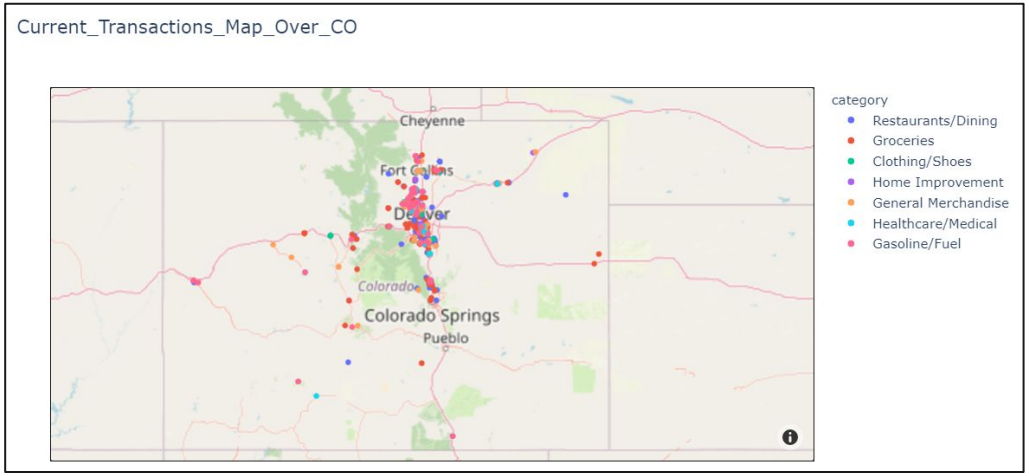
Location Based Recommendation System

Location Based Recommendation System

- Goal 2:
  - Based on the location of a merchant, other merchants in the vicinity are recommended to a user

Data

- For this part of the project only records in Colorado are selected.
- Since not all categories are equally represented in the data, only categories with greater than 150 records were selected.



Models

- To recommend similar merchants based on location the data needs to be clustered such that similar locations are within the same cluster

- KMeans, DBSCAN and HDBSCAN were used for this project

- **KMeans:**
  - Very simple algorithm for clustering but do require number of clusters needed as input
  - Doesn't cluster well the data is dense on some locations and sparse on other location
  - Didn't produce optimal clusters for this project and therefore wrong recommendations were generated
  - Not recommended for the client

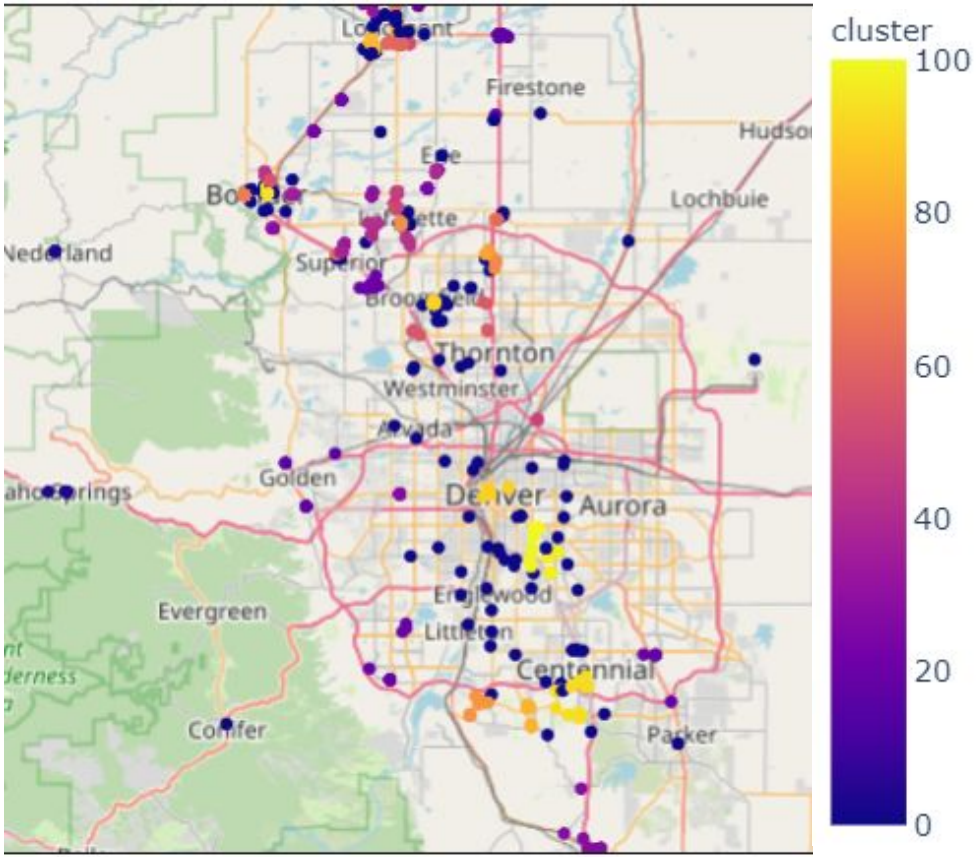
- **DBSCAN (Density Based Spatial Clustering of Application with Noise):**
  - Is perfect for a data with varying levels of density.
  - Requires epsilon(eps) and min_samples as inputs
  - The sklearn library does not contain a predict function for this hierarchical algorithm - limitation of hierarchical models since any new addition causes the process to start over
  - Since there is no predict function for DBSCAN in sklearn, this model was not used for this project

Models

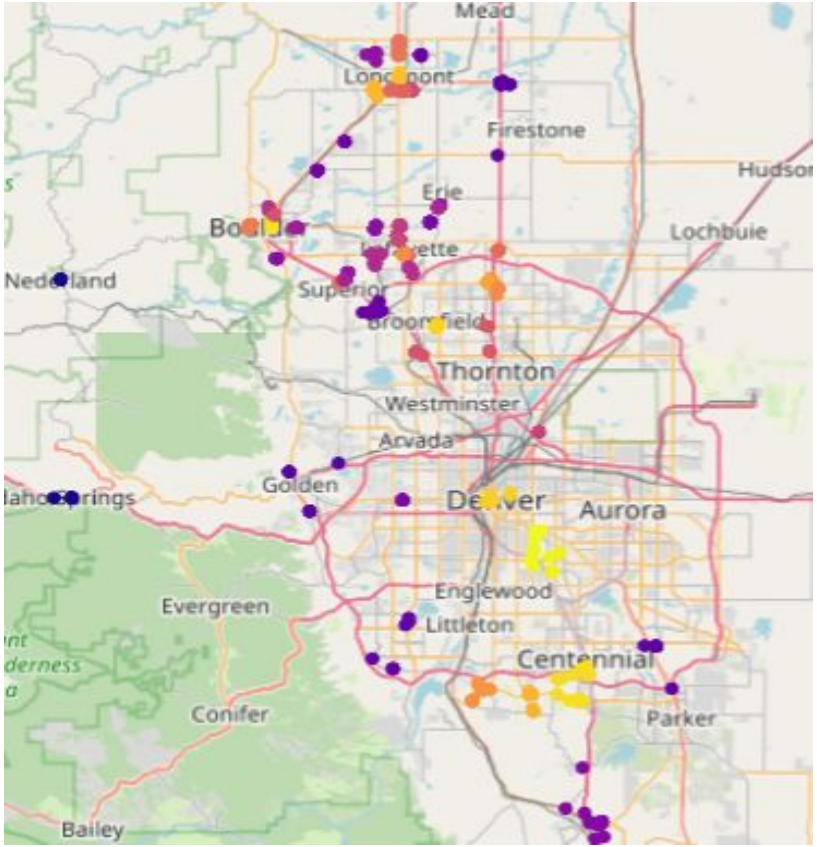
- **HDBSCAN (**Hierarchical DBSCAN)
  - Performs DBSCAN over varying values of eps
  - Unlike DBSCAN HDBSCAN finds clusters of varying densities
  - Hdbscan library was imported to apply it on the data
  - Only required input parameter that changes that clusters' configuration is min_cluster_size
  - It uses method approximate_predict to identify a cluster from a trained data for any new data point
- Steps
  - Identify the best min_cluster_size and apply hdbscan to create clusters (min_cluster_size=5)
  - Once clusters are identified, use approximate_predict with test data to find top 5 recommendations in the surrounding area.

Clustering with HDBSCAN - Zoomed In Map

With noise



Without noise



Clustering with HDBSCAN

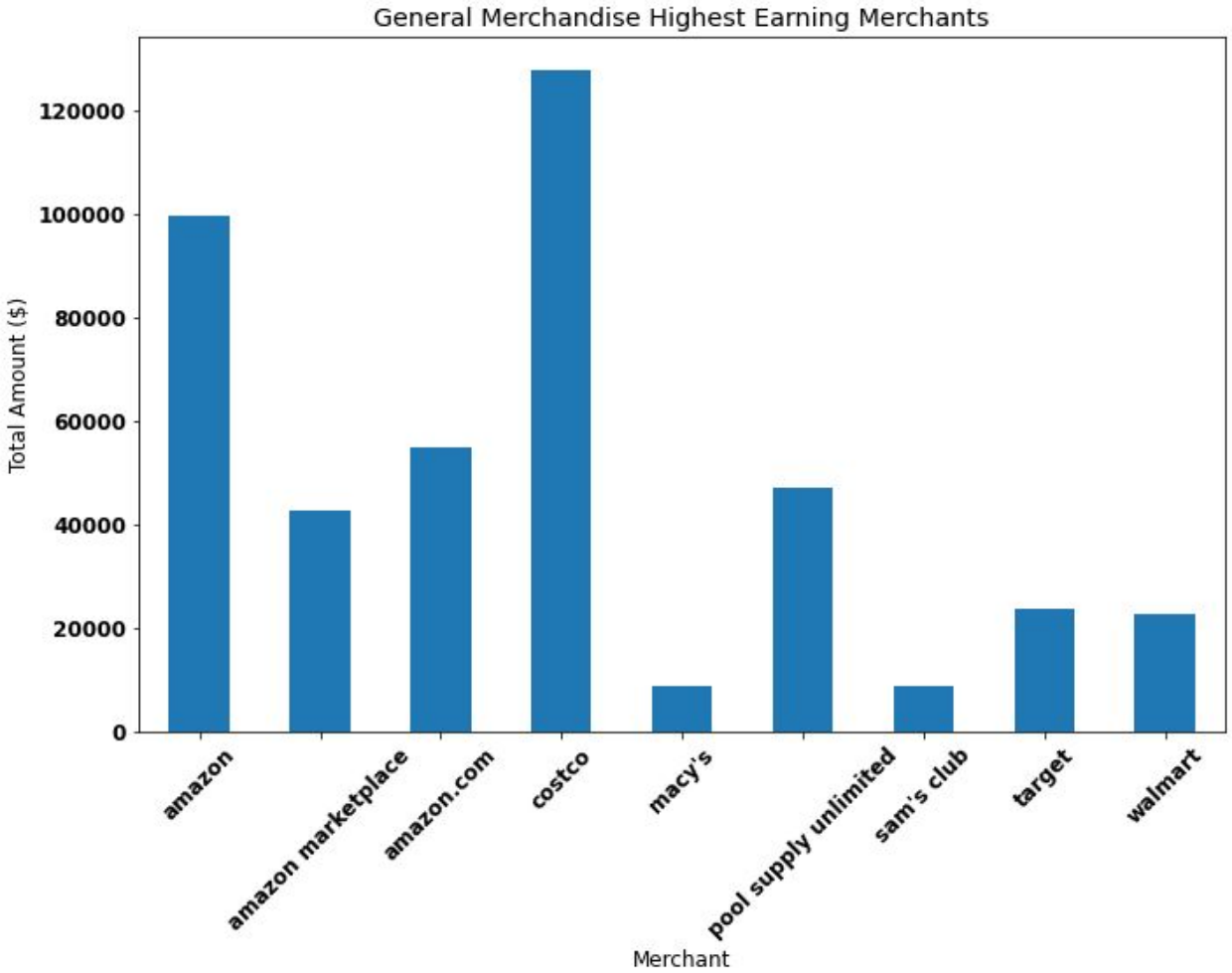
- Observations:
  - The parameter min_sample_size = 5 generated minimum number of noise data points
  - Even though overall the noise points were lower, there are many still existent within high density area of Denver and surrounding area
  - From the zoomed in versions of the maps it can be observed that while closer data points are clustered correctly, alot of points around those clusters were tagged as noise as well.
  - The cause of concern here is the data points between the Denver and Aurora area where it seems like that HBSCAN should have made better clusters or added more points around that area into existing clusters
  - IT IS NOT PERFECT BUT BETTER THAN KMEANS

Test and Result

- Arbitrarily chosen test points from the data set were provided to the model's approximate_predict. Following shows the result from one of those tests
- The data point was
    - (40.15403, -105.10333, 'Longmont', "mcdonald's")

	merchant	city	latitude	longitude
0	first watch	Longmont	40.15203	-105.09626
1	modern market	Longmont	40.15203	-105.09626
2	la mariposa	Longmont	40.15203	-105.09626
4	safeway fuel	Longmont	40.15203	-105.09626

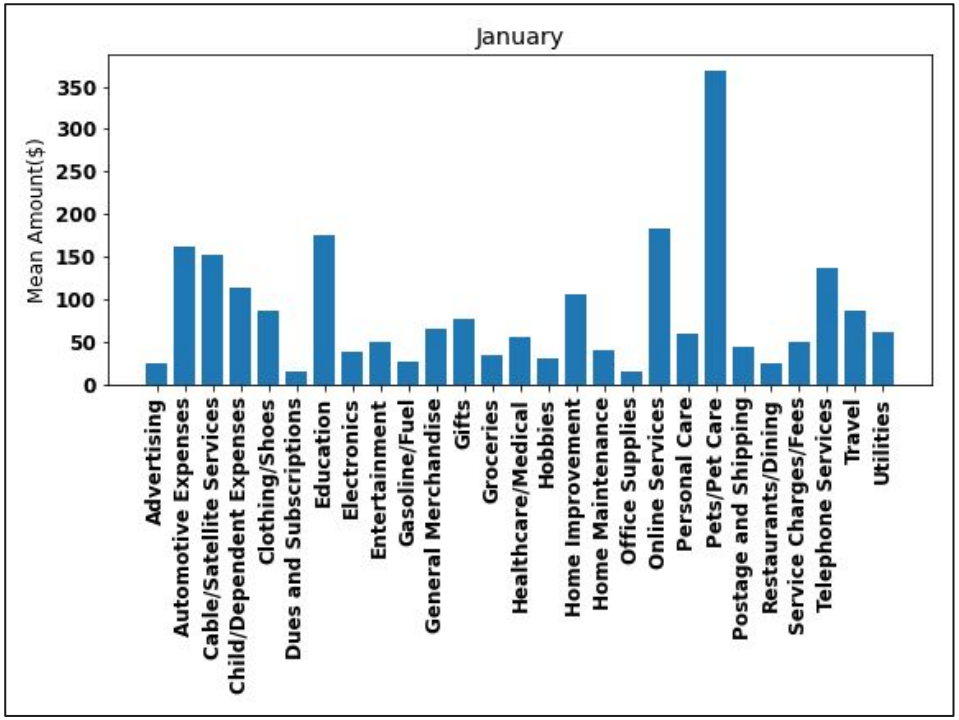
Backup Slides

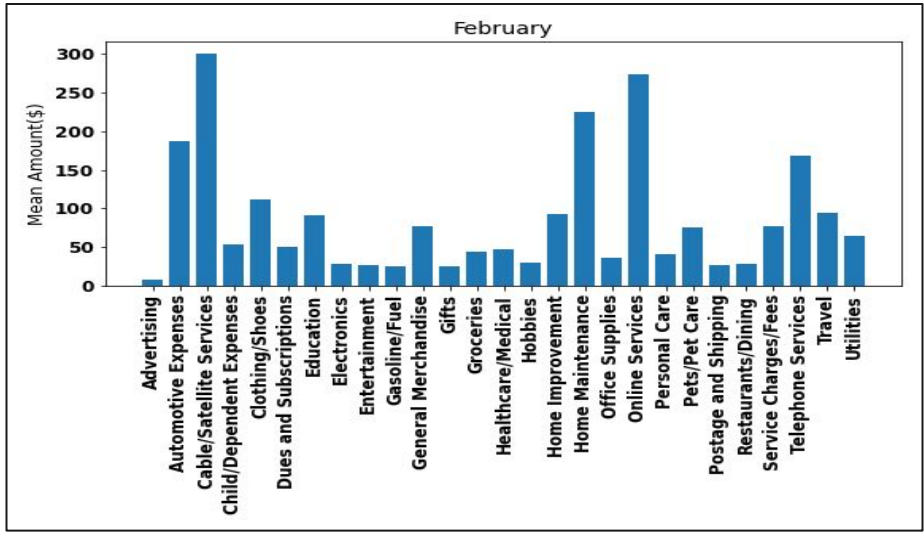


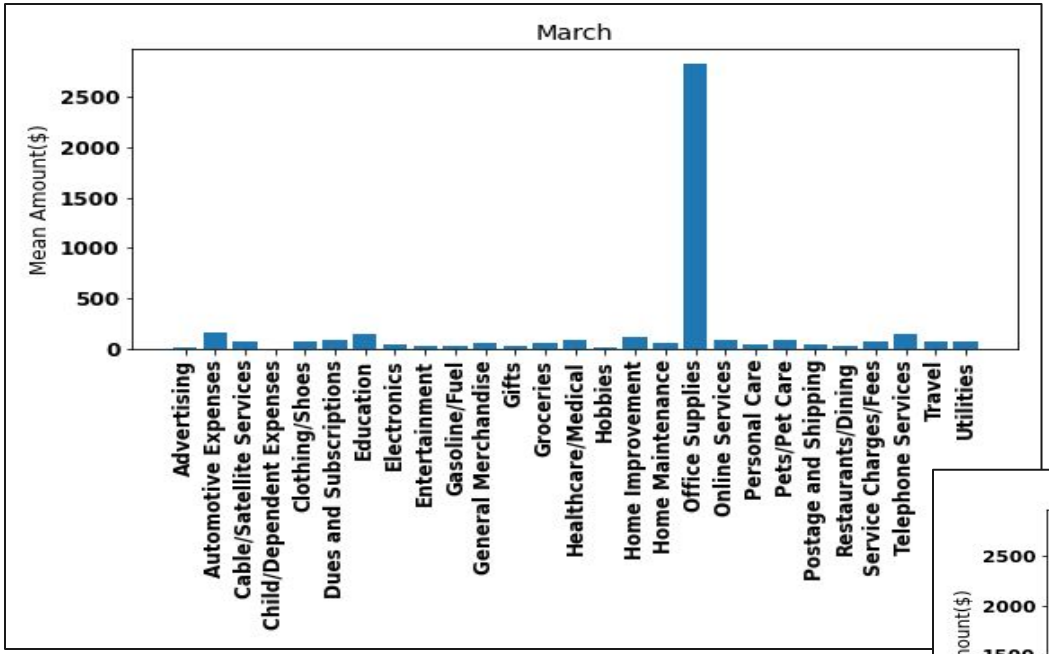
Users spent $128K at Costco (highest amount in the General Merchandise category)

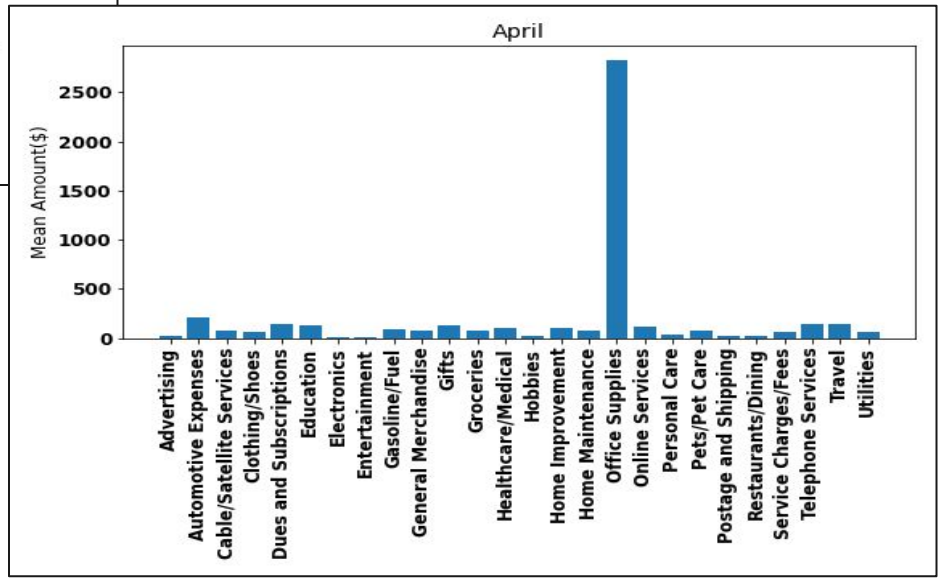
All the top names in this category are well known except for ‘pool supply unlimited’.

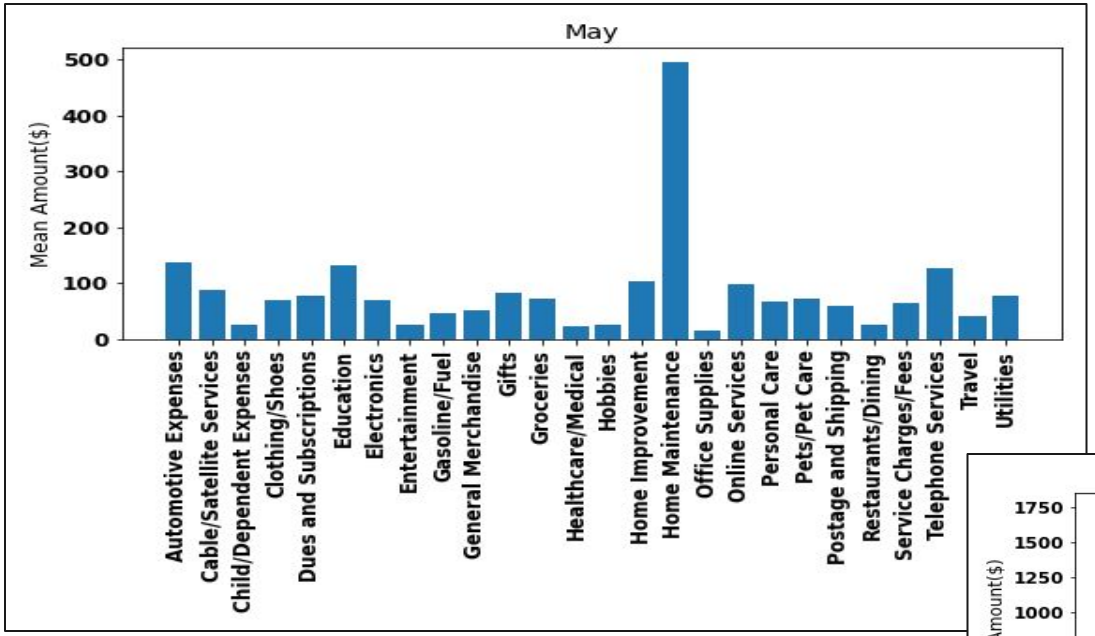
Upon investigations it was found that some high dollar amount in ‘pool supply unlimited’ transactions were duplicates.

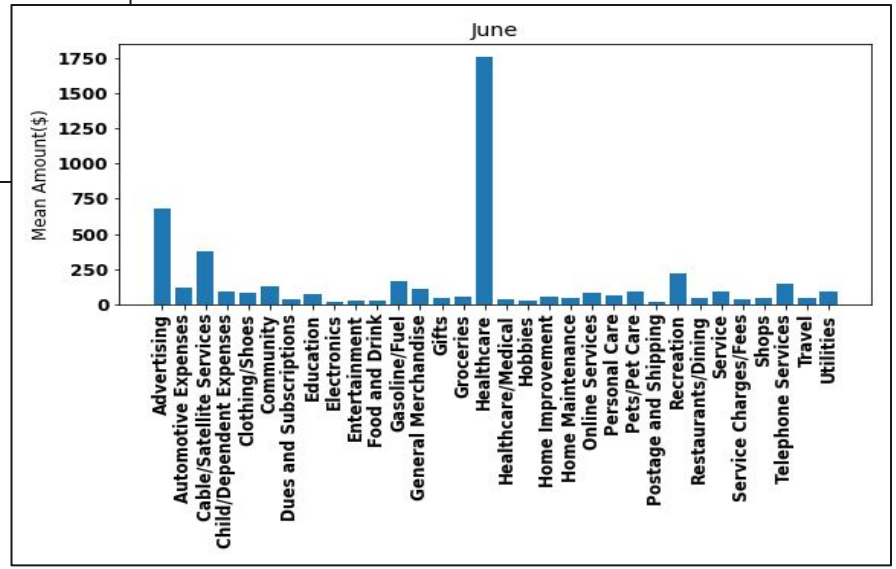

January
Mean Amount($)
0
50
100
150
200
250
300
350
Advertising
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Dues and Subscriptions
Education
Electronics
Entertainment
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Office Supplies
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Restaurants/Dining
Service Charges/Fees
Telephone Services
Travel
Utilities

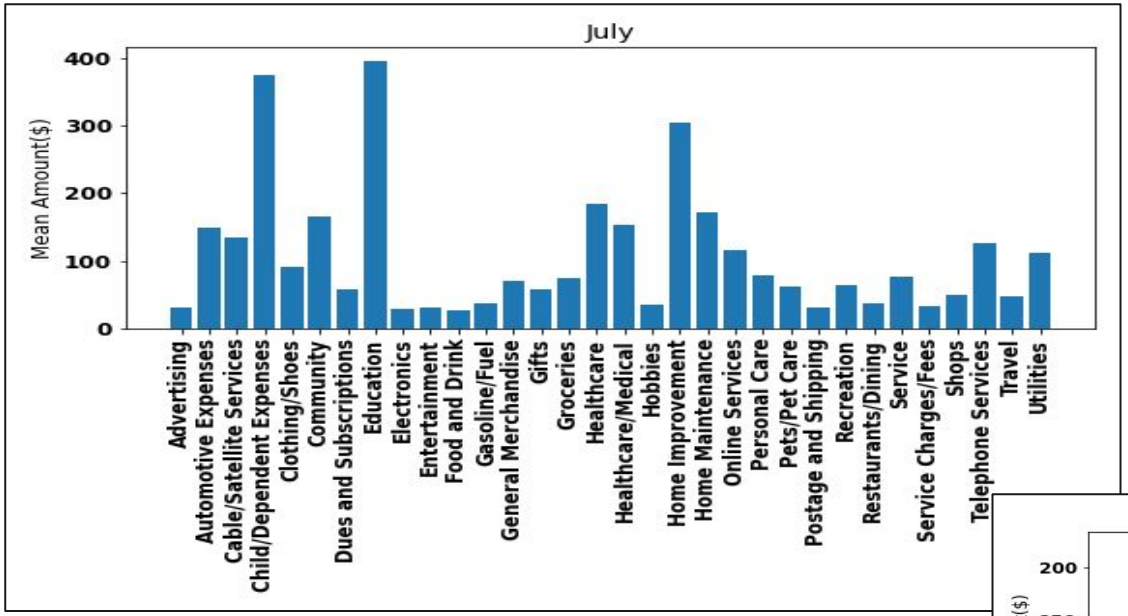

February
Mean Amount($)
0
50
100
150
200
250
300
Advertising
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Dues and Subscriptions
Education
Electronics
Entertainment
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Office Supplies
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Restaurants/Dining
Service Charges/Fees
Telephone Services
Travel
Utilities

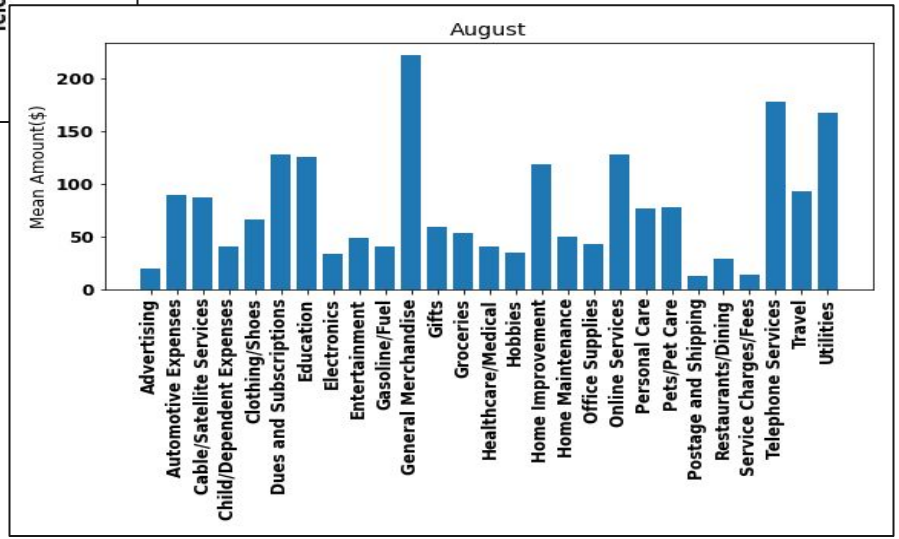

March
Mean Amount($)
0
500
1000
1500
2000
2500
Advertising
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Dues and Subscriptions
Education
Electronics
Entertainment
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Office Supplies
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Restaurants/Dining
Service Charges/Fees
Telephone Services
Travel
Utilities


April
Mean Amount($)
0
500
1000
1500
2000
2500
Advertising
Automotive Expenses
Cable/Satellite Services
Clothing/Shoes
Dues and Subscriptions
Education
Electronics
Entertainment
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Office Supplies
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Restaurants/Dining
Service Charges/Fees
Telephone Services
Travel
Utilities


May
Mean Amount($)
500
400
300
200
100
0
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Dues and Subscriptions
Education
Electronics
Entertainment
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Office Supplies
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Restaurants/Dining
Service Charges/Fees
Telephone Services
Travel
Utilities


June
Mean Amount($)
1750
1500
1250
1000
750
500
250
0
Advertising
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Community
Dues and Subscriptions
Education
Electronics
Entertainment
Food and Drink
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Recreation
Restaurants/Dining
Service
Service Charges/Fees
Shops
Telephone Services
Travel
Utilities


July
Mean Amount($)
0
100
200
300
400
Advertising
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Community
Dues and Subscriptions
Education
Electronics
Entertainment
Food and Drink
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Recreation
Restaurants/Dining
Service
Service Charges/Fees
Shops
Telephone Services
Travel
Utilities


August
Mean Amount($)
0
50
100
150
200
Advertising
Automotive Expenses
Cable/Satellite Services
Child/Dependent Expenses
Clothing/Shoes
Dues and Subscriptions
Education
Electronics
Entertainment
Gasoline/Fuel
General Merchandise
Gifts
Groceries
Healthcare/Medical
Hobbies
Home Improvement
Home Maintenance
Office Supplies
Online Services
Personal Care
Pets/Pet Care
Postage and Shipping
Restaurants/Dining
Service Charges/Fees
Telephone Services
Travel
Utilities

Location Based Recomm_System - Code for building the recomm_df

In [34]:
```
recomm_df = df2_co_topcat.copy()
recomm_df = (recomm_df.groupby(['cluster', 'merchant']).agg({'latitude' : 'first',
                                                             'longitude' : 'first',
                                                             'city' : 'first',
                                                             'category' : 'first',
                                                             'cluster' : 'count'})
          .rename({'cluster' : 'cluster_count'},axis=1).reset_index()
          .sort_values(['cluster', 'cluster_count'], ascending = [True, False])
          .drop('cluster_count', axis=1))

recomm_df.head()
```

Out[34]:

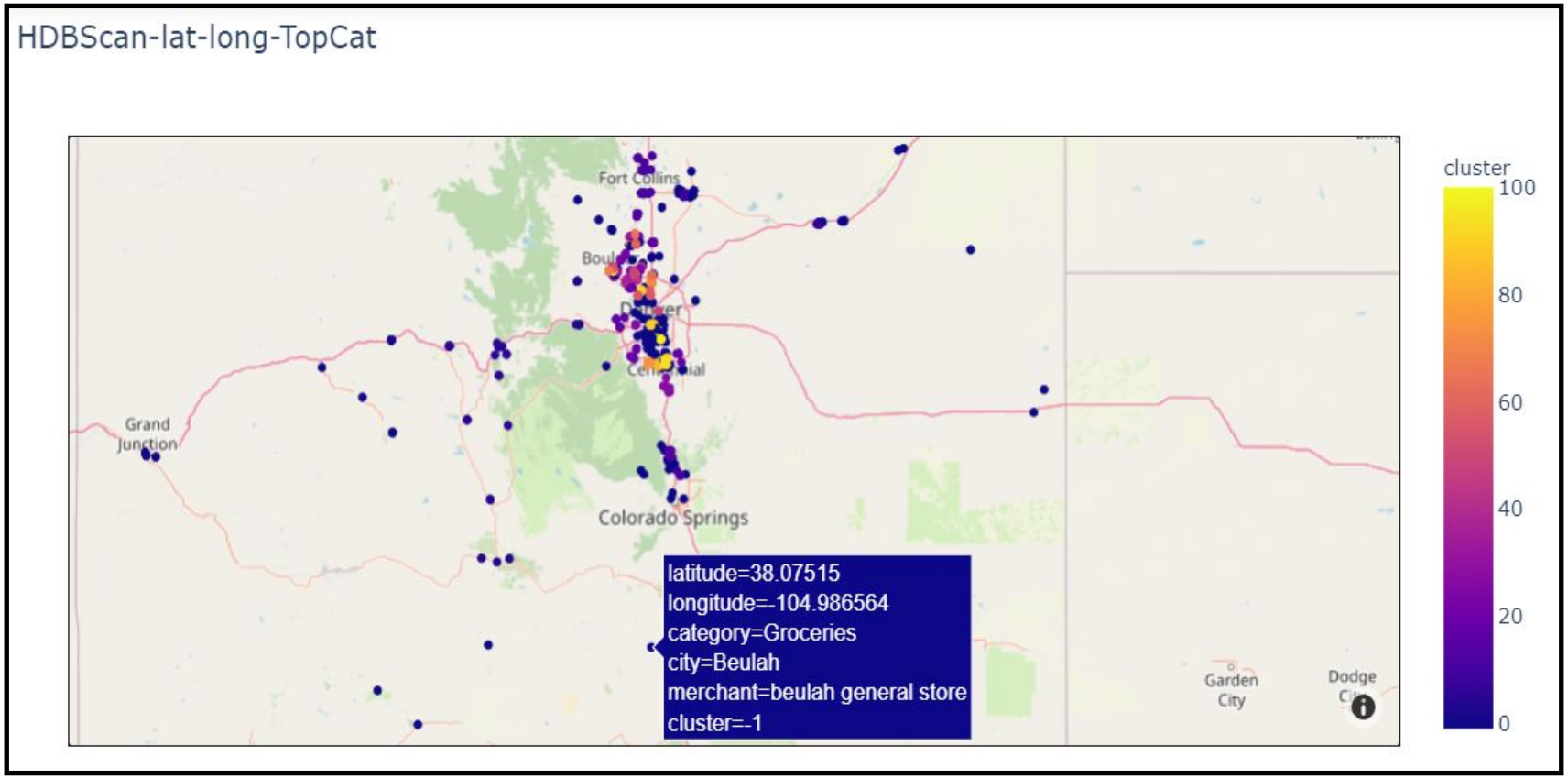
	cluster	merchant	latitude	longitude	city	category
2	-1	ace hardware	39.768741	-105.00495	Denver	Home Improvement
3	-1	altar'd state	40.017740	-105.25541	Boulder	Clothing/Shoes
4	-1	andy's frozen custard	39.646900	-105.02530	Sheridan	Restaurants/Dining
5	-1	arby's	39.596080	-104.88609	Greenwood Village	Restaurants/Dining
6	-1	barking dog cafe	40.224548	-105.27076	Lyons	Restaurants/Dining

Location Based Recomm_System - Code for Recommendation Engine

```
def recommend_co_merchants_hdb(df,lat,long,city,merchant,cluster_object):
    # Predict the cluster for longitude and latitude provided
    test_labels, strengths = hdbscan.approximate_predict(cluster_object, [[lat,long]])
    predicted_cluster=test_labels[0]
    print('Predicted cluster for this lat/long combination is: '  + str(predicted_cluster))
    print("_____________________________________________________________________________")

    if predicted_cluster==-1:
        return ('No merchants close by')
    # Get the best merchant in this cluster
    else:
        pop_merch_recomm_df=(df[df['cluster']==predicted_cluster].iloc[0:5][['merchant','city','latitude','longitude']])
        pop_merch_recomm_df=pop_merch_recomm_df.reset_index(drop=True)
        mask = (pop_merch_recomm_df.merchant==merchant) & (pop_merch_recomm_df.latitude==lat) & \
               (pop_merch_recomm_df.longitude==long)
        print ('Since you are currently in '+ city.capitalize() + ' ' + 'at ' + \
               merchant.capitalize() + ', how about you visit these merchants around this area? ')
    return pop_merch_recomm_df[~mask]
```

Visualization of data with noise present



Visualization of data without noise