Aditya Kumar
Vivek Vasireddy
December 18, 2022

397A Final Project Writeup

Group Members and Work:

Vivek Vasireddy and **Aditya Kumar**

How We Worked Together: Had set times in which we met together and worked on the code together in order to achieve our goal. Had an even split of doing the work in order to achieve the completion of this project alongside both members attending Office Hours and reaching out to the TA for help.

Dataset:

The dataset we used for this project was the **social_capital_zip.csv**. This csv was among the pre-approved datasets listed in the project guidelines document. We chose this dataset because we wanted to examine how various socioeconomic factors impacted social capital and economic connectedness.

In this dataset, the main processing we did was just setting the index to the "zip" column as that was the unique identifier and also just cleared off all the NA values, as that would be a detriment to our project if we had kept them in as we went along from plotting the visualization and even to the predictive modeling.

Here's how we achieved that goal:

```
# Reading in the Data and checking it to ensure it's output is correct
socCap_df = pd.read_csv("social_capital_zip.csv")

# Setting index to zip for future use in Phase 2
socCap_df.set_index('zip', inplace=True)
```

Image 1: Reading & Setting Index

```
# Dropping NA Values in the dataframe such that we have points that exist to do our analysis on.
socCap_df = socCap_df.dropna()
```

Image 2: Dropping NA values

In the first image, we read in the dataset and using the prior knowledge we had gained from class, we were able to set the index to the zip column as aforementioned. This

would allow us to uniquely identify each and every input as each zip code in the USA that is not duplicated. In the 2nd image, we just used the same prior knowledge we had acquired in class in order to drop the NA values to clear our dataset. Doing this resulted in a drop from the original dataset containing 23028 values to going down to 14269 values.

Phase 1: Basic Visualizations

To start off, we first looked to create five basic visualizations. Given that our data we looked to examine wasn't continuous in any way, the scatter plot was our best way forward. Doing so led us to examine the five particular variables of: ***ec_se_zip, ec_zip, volunteering_rate_zip, civiv_organizations_zip*** and ***clustering_zip.*** Examining how the overall variables we pointed out before was done by plotting them against the *pop2018* variable which showed the overall population in the year 2018. Our main focus was to examine the overall civic engagement and as such those five variables plotted against the population was the best choice for us. The scatter plots were present to not only introduce us to the dataset and allowed us to become familiar with it, but it also helped us to quickly visualize the trend in data.

One area of difficulty that we had experienced was that in our scatterplots, whilst being able to see the overall trend, we couldn't exactly figure out how to create a trendline. In previous classes, such as Informatics 248 where we worked in R, when we created a plot we were able to create a graph with an abline that would help to model the overall linear trend of the data for quick use. It is shown below.

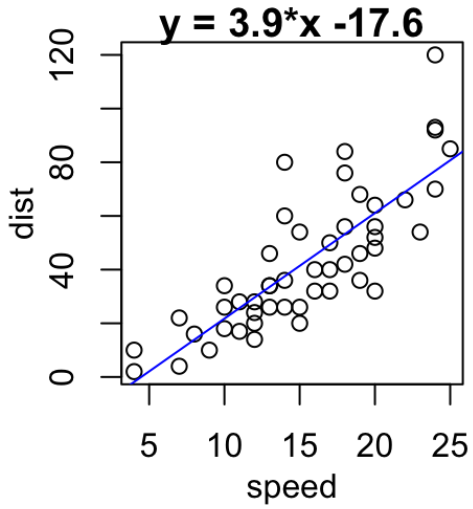
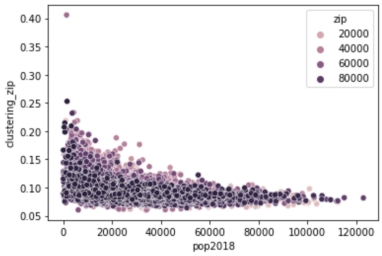


Image 3: Usage of abline in R

However, we weren't able to combat this as we had attempted a solution which was that using the different plots of **regplot** and **lmplot** as looking it up had shown us that those were the only two plots that had been built in with an abline. The main issue with them against using the basic sns scatterplot was that in the graphs, they looked quite mixed up and didn't allow for us to visualize our data quite strongly enough.

```
# Plotting Basic Visualizations

plot5 = sns.scatterplot(data = socCap_df, x = "pop2018", y = "clustering_zip", hue = "zip")
plot5 = sns.lmplot(data = socCap_df, x = "pop2018", y = "clustering_zip")
```



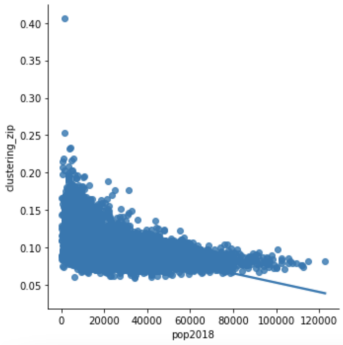


Image 4: Basic Visualizations (Comparing sns.scatterplot to sns.lmplot)

Now, as we looked at this, we were able to see that the **hue** attribute in sns.scatterplot allowed us to better view the data as it'd group by zip codes. The other issue is that the data in the sns.lmplot was just so jumbled that it didn't have any main effect on allowing us to further examine the data.

Overall Takeaways from Phase 1: Basic Visualizations - In our data, we were able to see that the majority of the variables that we plotted had an overall negative effect when being graphed out.

Phase 2: Clustering

As we looked to create a clustering model, the main goal for this section was to create a model that, based on the zipcode as it was the unique identifier for us, had hoped to create a model that would best output our results for this section. Starting off we just created a function that would print out all the columns in our data to best display the overall columns that we were able to choose. The four that we ended up choosing are displayed below:

```
# Creating Columns we plan to create clusters on

clust_columns = ["ec_zip", "volunteering_rate_zip", "civic_organizations_zip", "clustering_zip"]

# Selecting our number of clusters

num_clusts = 4

# Selecting the linkage type for our clustering

linkage_type = "complete"

df_cluster = socCap_df[clust_columns].dropna()
```

Image 5: Selection of Clustering Columns and Linkage Type

This also includes our linkage type, which was complete and dropped all the NA values which were already done before, but just there as a precaution. We chose those four variables, as they were the ones we had examined in the Phase 1 of Basic Visualization and we felt were the most important to us to use for clustering in this section of the project.

Then by using the overall techniques we learnt in class, we were able to write a couple for loops in order to print out our results for clustering.

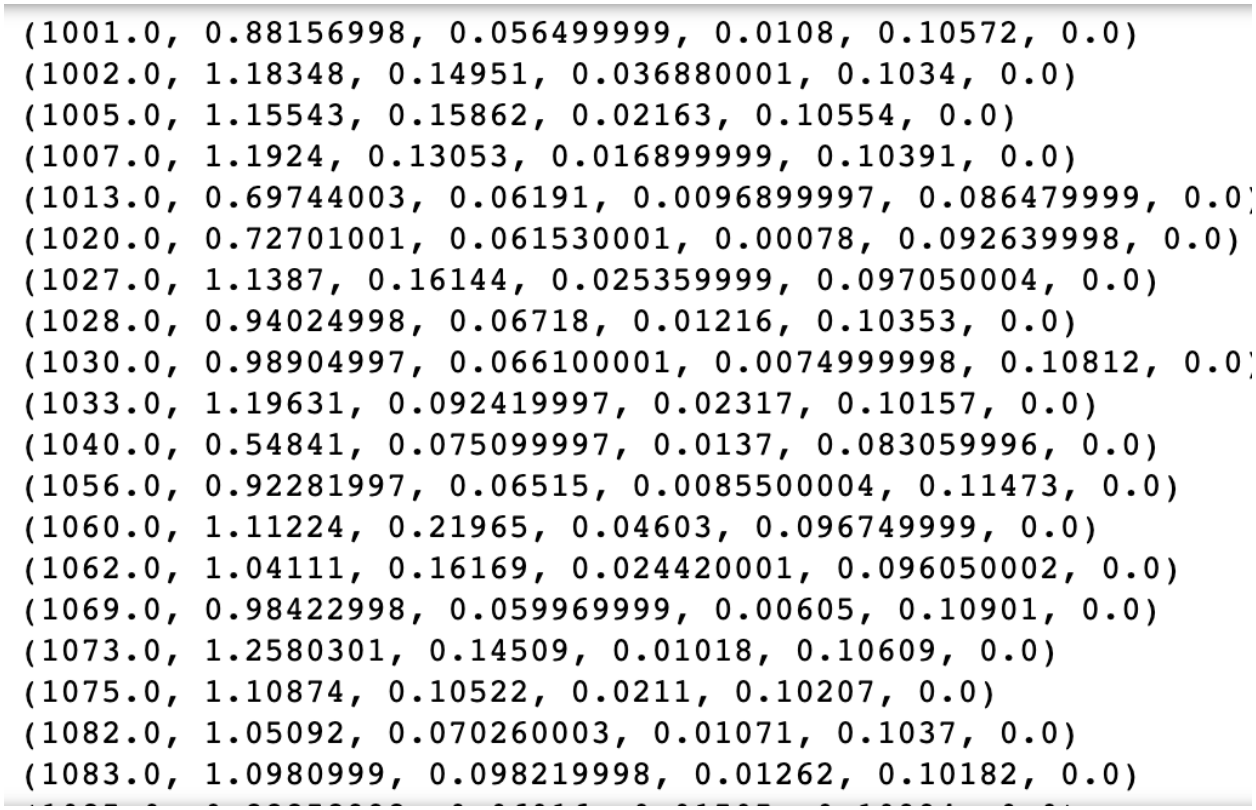


Image 6: Result of Clustering

Form of: [“zip, "ec_zip", "volunteering_rate_zip", "civic_organizations_zip", "clustering_zip"]

The one issue that we ran into in this phase was no matter how hard we tried, there would always be a 0.0 value in the last column. This is something we also noticed within the class dataset and as such we assumed it was normal.

From our clustering, we were able to better visualize our related data that we had chosen to be the most useful for examining amongst the civic engagement and economic connectedness columns.

Phase 3: Predictive Modeling

We first made a categorical class label called “volunteering_rate”. After we set a range of 0.075 as we felt that was the best for the data, it was split up into a categorical value of 0 or 1. We used this as when we aimed to create our decision tree, having a binary classifier would split our dataset into the two parts that we needed. From then on we created a 80/20 split between our test and training datasets as that was the bounds we found to be right.

The model we decided to learn and apply was a Decision Tree model. One of the reasons we decided to use this kind of model was because we had already got rid of all the NA values before, so we were confident that this technique would be able to handle our data.

One of the main issues we had was that after creating the datasets and finding out the accuracy and prediction of the models, we found out that no matter what we pretty much got around 1.0. The one exception to this was when our *test_size* was 90% and our range for the formula was set to < 0.050. This resulted in our accuracy being 0.999 which in effect was a value of 1.0. The result is shown below:

```
[1 1 1 ... 0 0 0]
zip
17019    1
21631    1
45334    1
29040    0
39140    0
        ..
37616    1
55336    1
67210    0
72764    0
75233    0
Name: Volunteering Rate ZIP Prediction, Length: 12843, dtype: int64
Accuracy: 0.9987541851592308
Precision: 1.0
```

Image 7: High Test Data and Low Bound - Accuracy and Precision

In doing so, we assumed that we had done something wrong as we felt that no matter what the values of accuracy and precision were always one if not very close to one. Reaching out to the TA was our best bet and talking to Siddarth during Office Hours allowed us to test many different values in our data to reach the conclusion that due to overfitting and having such data, that we had not made any mistakes. The final results for this statistical association are shown below:

```
[1 1 1 ... 1 1 1]
zip
39183    1
91789    1
38834    1
48611    0
72634    0
        ..
80908    1
31079    1
98812    1
25621    1
32696    1
Name: Volunteering Rate ZIP Prediction, Length: 2854, dtype: int64
Accuracy: 1.0
Precision: 1.0
```

Image 8: Final Accuracy and Precision *(Bound of > 0.075 and 20% Train Data)*

As our data was so close together, we felt that we had done a good job in choosing a variable that best modeled our data given by the accuracy and precision numbers shown.

Phase 4: Statistical Association

For our 4th concept, we aimed to complete a statistical association test on the *volunteering_rate_zip* variable as it has been mentioned a lot of times before that our focus was on the overall civic engagement in this dataset. To achieve this, we first created another column in our data that was the predictor which was a binary predictor being that of 0 or 1. In our case, if the value was > 0.075 then it was listed as a 0, else it was a 1. We thought this was the best as when we aimed to plot the data (talked about later on) , grouping them into two categories split by a value was the best choice we could make to best represent our data. The creation of the function is shown below:

```
# Creating Function for Association
socCap_df['VolunteeringRateZipPred'] = socCap_df['volunteering_rate_zip'].apply(lambda x: "0" if x > 0.075 else "1")

socCap_df
```

Image 8: Creation of Function for the Association Test

VolunteeringRateZipPred
1
0
0
0
1
...
0
0
0
0
0

Image 9: Result of said Function in new column

By achieving this result, we were able to then use our data in the creation of a stripplot as well as showing the overall chi-sq test which contained a lot of useful values such as the chi squared value, the p-value and degrees of freedom.

```
contingency = pd.crosstab(socCap_df["VolunteeringRateZipPred"], socCap_df["civic_organizations_zip"])
print(contingency)

# Building a chi-sq test

chisq, pval, dof, expected = st.chi2_contingency(contingency)
print(chisq, pval, dof, expected)

g = sns.stripplot(data = socCap_df, x = "VolunteeringRateZipPred", y = "civic_organizations_zip")
```

```
civic_organizations_zip  -0.000740  -0.000580  -0.000550  -0.000506  \
VolunteeringRateZipPred
0                                0          0          0          1
1                                1          1          1          0

civic_organizations_zip  -0.000460  -0.000360  -0.000340  -0.000270  \
VolunteeringRateZipPred
0                                0          0          0          0
1                                1          1          1          1

civic_organizations_zip  -0.000100  -0.000080  ...   0.277320   0.285340  \
VolunteeringRateZipPred                        ...
0                                1          0  ...          1          0
1                                0          1  ...          0          1

civic_organizations_zip   0.285660   0.321340   0.335690   0.383560  \
VolunteeringRateZipPred
0                                0          1          1          1
1                                1          0          0          0

civic_organizations_zip   0.406990   0.414730   0.434260   0.589270
VolunteeringRateZipPred
0                                1          1          0          1
1                                0          0          1          0

[2 rows x 3576 columns]
4273.366853319159 3.5396562658146838e-15 3575 [[0.40100918 0.40100918 0.40100918 ... 0.40100918 0.40100918 0.4010091
8]
 [0.59899082 0.59899082 0.59899082 ... 0.59899082 0.59899082 0.59899082]]
```

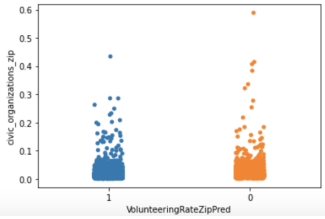


Image 10: Chi Squared Test and Stripplot.

As shown here, the overall data for this displays that the overall *civic_engagement_zip* does not vary that much as shown by the stripplot after grouping the *volunteering_rate_zip* values. We were also able to ensure that this was a significant test as the p-value for this was $3.52*10^{-15}$ which is way less than the value of 0.05, therefore indicating a significant test.

Phase 5: Geocoding

Knowing that our data contained the zip codes of multiple counties across the US, we had hoped to create a geocoding map as we did in class, since we thought that it was some of the most enjoyable things we learned in the class this semester. However, we weren't able to successfully complete this phase of the project due to a lack of experience on how to complete geocoding without the presence of latitude or longitude values alongside the physical addresses. We tried to complete this section of the project by mimicking the similar steps we had taken in class when we did the geocoding. We created a duplicate dataset from our main data that we had set earlier on. In doing so, we also set the index column to "zip" and built the data with the column we wanted to graph, which was the *volunteering_rate_zip* column. However, when running this code chunk the error always persisted which was that there was a KeyError: 'cluster'. The code is shown below alongside the error.

```
# Geoencoding

geo_df = socCap_df.copy()
geo_df['zip'] = geo_df.index

peers = geo_df.groupby('cluster')['zip'].count().reset_index()
peers.rename(columns = {"zip": "peers"}, inplace = True)

geo_df = pd.merge(geo_df, peers, on = 'cluster')
geo_df = gpd.GeoDataFrame(geo_df)

geo_df.explore("cluster", tooltip=["zip", "volunteering_rate_zip", "cluster", "peers"], categorical = True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [8], in <cell line: 5>()
      2 geo_df['zip'] = geo_df.index
      4 # Add a column with the cluster size (peers) for each town
----> 5 peers = geo_df.groupby('cluster')['zip'].count().reset_index()
      6 peers.rename(columns = {"zip": "peers"}, inplace=True)
      8 geo_df = pd.merge(geo_df, peers, on = 'cluster')

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/frame.py:7712, in DataFrame.groupby(self, by, axis, level, as_index, sort, group_keys, squeeze, observed, dropna)
   7707 axis = self._get_axis_number(axis)
   7709 # https://github.com/python/mypy/issues/7642
   7710 # error: Argument "squeeze" to "DataFrameGroupBy" has incompatible type
   7711 # "Union[bool, NoDefault]"; expected "bool"
-> 7712 return DataFrameGroupBy(
   7713     obj=self,
   7714     keys=by,
   7715     axis=axis,
   7716     level=level,
   7717     as_index=as_index,
   7718     sort=sort,
   7719     group_keys=group_keys,
   7720     squeeze=squeeze,  # type: ignore[arg-type]
   7721     observed=observed,
   7722     dropna=dropna,
   7723 )

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/groupby/groupby.py:882, in GroupBy.__init__(self, obj, keys, axis, level, grouper, exclusions, selection, as_index, sort, group_keys, squeeze, observed, mutated, dropna)
    879 if grouper is None:
    880     from pandas.core.groupby.grouper import get_grouper
--> 882     grouper, exclusions, obj = get_grouper(
    883         obj,
    884         keys,
    885         axis=axis,
    886         level=level,
    887         sort=sort,
    888         observed=observed,
    889         mutated=self.mutated,
    890         dropna=self.dropna,
    891     )
    893 self.obj = obj
    894 self.axis = obj._get_axis_number(axis)

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/groupby/grouper.py:882, in get_grouper(obj, key, axis, level, sort, observed, mutated, validate, dropna)
    880         in_axis, level, gpr = False, gpr, None
    881     else:
--> 882         raise KeyError(gpr)
    883 elif isinstance(gpr, Grouper) and gpr.key is not None:
    884     # Add key to exclusions
    885     exclusions.add(gpr.key)

KeyError: 'cluster'
```

Image 11: Geocoding Error

We tried to solve this problem by doing some google searching alongside going back into our geocoding lab however these attempts were unsuccessful. We left this here as we had spent a lot of time attempting to complete this but were not successful and the fact that we were able to achieve four other sections completely.

Future Exploration:

As previously stated in Phase 5, we were unsuccessful in attempting to complete the overall geocoding section. As such, if we were able to continue on that would be step 1 of the continuation which would be to complete said section.

From them on, we would look to further bolster our analysis by comparing many other variables. Our main focus was that we looked to mainly examine the civic engagement statistic as seen by the heavy analysis of the *volunteering_rate_zip* and *civic_organizations_zip* variables. We would like to examine how the overall effect of economic connectedness would play into something like this, given the dataset we had.