FinalProjectJKRUSE

August 4, 2024

1 Exploring Health and Safety of Los Angeles County Restuarants

[174]:
```
# import packages
import pandas as pd
from PIL import Image
import geopandas as gpd
import numpy as np
import folium
from IPython.display import display
import statsmodels.api as sm
import re
```

1.1 Introduction

Los Angeles county California is the largest US county by population and is home to about 10 million people from a diverse range of ethnic and socioeconomic backgrounds. This diverse population of people produces an equally diverse culture of food which can be found in the tens of thousands of restaurants operating in the area. In this project I will look at the LA County health inspection data to determine the overall safety of restaurants in the area, as well as to see if different segments of the population face greater health and safety risks. Do certain geographic locations within LA County have better/worse food safety? Are lower income areas disadvantaged by lower food safety? This inforamation could be useful to the the LA County department of public health in the allocation of education, inspections and resources. It could also be helpful for individuals living in LA County to better understand the food they eat and to better know the risks they may face doing so. For this project, data was taken from this Kaggle dataset from about 200,000 health inspections between July 2015 and December 2017.

1.2 Data Analysis

[175]:
```
# Load in data
inspections_df = pd.read_csv('inspections.csv')
violations_df = pd.read_csv('violations.csv')
```

[176]:
```
# Examine the shape of the data
print(inspections_df.shape)
print(violations_df.shape)
```

```
(191371, 20)
(906014, 5)
```

The inspections dataframe holds rows corresponding to 191371 health inspections while the violations dataframe holds rows corresponding to 906014 health code violations from those inspections. There are 20 columns in inspections_df while only 5 columns in violations_df as shown below:

[177]:
```
pd.set_option('display.max_columns', None)
# And visualize the columns
inspections_df.head(1)
```

[177]:
```
  activity_date employee_id     facility_address  facility_city facility_id  \
0    2017-05-09   EE0000593  17660 CHATSWORTH ST  GRANADA HILLS   FA0175397

                facility_name facility_state facility_zip grade   owner_id  \
0  HOVIK'S FAMOUS MEAT & DELI             CA        91344     A  OW0181955

                       owner_name                           pe_description  \
0  JOHN'S FAMOUS MEAT & DELI INC.  FOOD MKT RETAIL (25-1,999 SF) HIGH RISK

   program_element_pe                program_name program_status  record_id  \
0                1612  HOVIK'S FAMOUS MEAT & DELI         ACTIVE  PR0168541

   score serial_number  service_code service_description
0     98     DAHDRUQZO             1  ROUTINE INSPECTION
```

[178]:
```
violations_df.head(1)
```

[178]:
```
   points serial_number violation_code  \
0       1     DAJ5UNMSF           F044

                              violation_description   violation_status
0  # 44. Floors, walls and ceilings: properly bui…  OUT OF COMPLIANCE
```

Some background information: Inspections start with 100 points and any incurred violations result in a loss of one or more points - A: 90-100, B: 80-90, C: 70-80. Scoring below 70 as well as the accumulation of multiple major violations can result in immediate reinspection or temporary/permanent closure.

Let's look at unique values for our `score` variable. We can see that scores range in the 50s up to 100, though not all possible scores in the 50s and 60s are contained in the data.

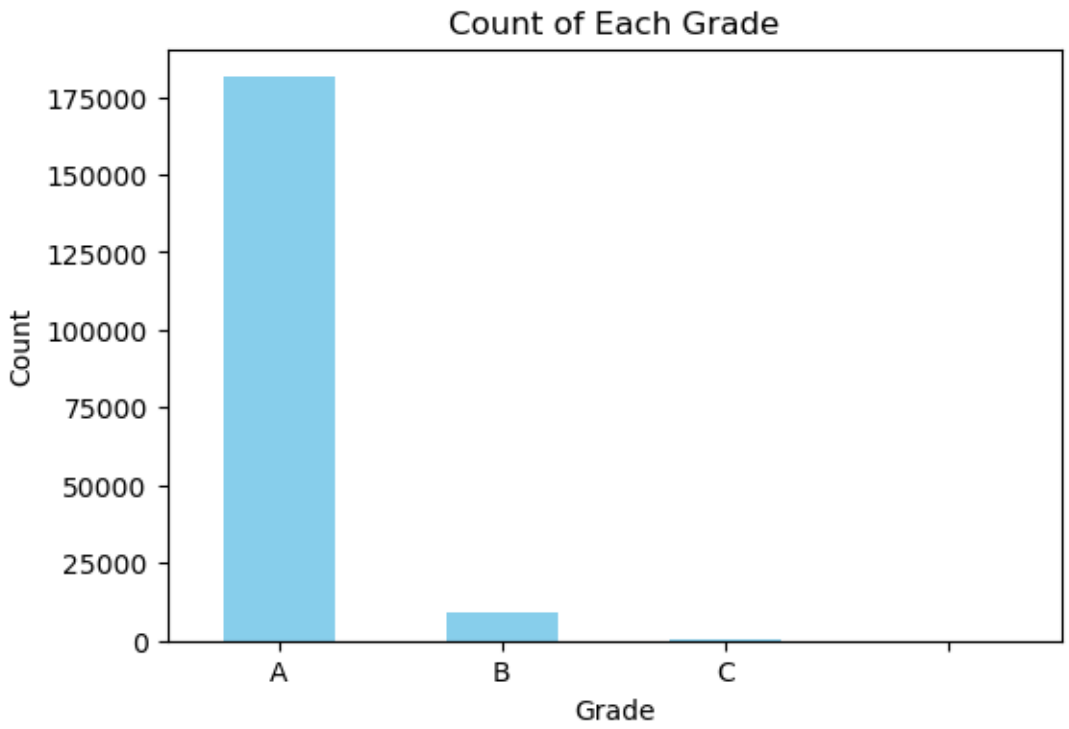
[179]:
```
print(sorted(inspections_df['score'].unique(), reverse=True))
inspections_df['grade'].unique()
```

```
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82,
81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62,
60, 59, 58, 57, 55, 54]
```

[179]:
```
array(['A', 'B', 'C', ' '], dtype=object)
```

Scores recieving less than an 'A' grade are uncommon, with even lower scores being exceedingly rare

[180]:
```
import matplotlib.pyplot as plt

# Count occurrences of each grade
grade_counts = inspections_df['grade'].value_counts()

# Plot the counts
plt.figure(figsize=(6, 4))
grade_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Grade')
plt.ylabel('Count')
plt.title('Count of Each Grade')
plt.xticks(rotation=0)  # Rotate x-axis labels if needed
plt.show()
```



Given that the proportion of scores in the 'A' range is so high, it may be difficult to find patterns in the prevelance of low scores.

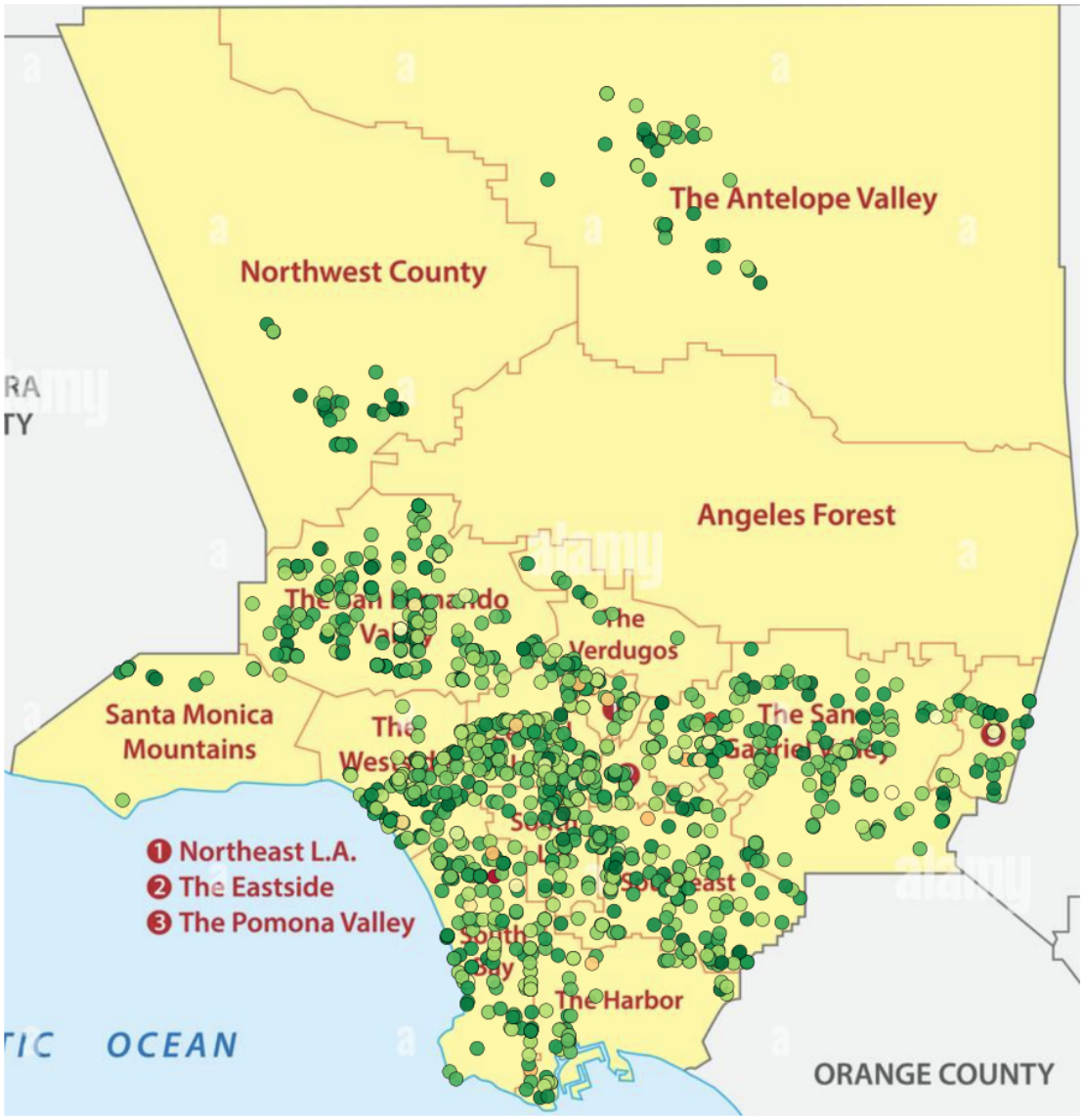
Because I wanted to visualize the health scores geographically, I tried to convert the `facility_address` and `facility_city` fields to coordinates. I was able to do this with some success. However, I struggled to do this using free resources. With the OpenCage free trial I was

able to convert the addresses from about 1500 randomly selected inpection entries to geographic coordinates.

Here is how these 1500 inspections look on a map of LA County when colored by `score`

[181]:
```
# Load the data
samp_with_coords = pd.read_csv('sample_with_coordinates.csv')

# Estimated long/lat coordinates of LA County's most north/south and east/west␣
 ↪points
north = 34.822719
south = 33.703727
east = -117.645524
west = -118.944880

# Load the image
image_path = 'lacountyimage.png'
image = Image.open(image_path)

# Convert latitude and longitude to pixel coordinates
def lat_to_pixel(lat, north, south, height):
    return height - (lat - south) / (north - south) * height

def lon_to_pixel(lon, west, east, width):
    return (lon - west) / (east - west) * width

# Calculate image dimensions
width, height = image.size

# Add x_pixel and y_pixel columns to the DataFrame
samp_with_coords['x_pixel'] = samp_with_coords['longitude'].apply(lon_to_pixel,␣
 ↪args=(west, east, width))
samp_with_coords['y_pixel'] = samp_with_coords['latitude'].apply(lat_to_pixel,␣
 ↪args=(north, south, height))


# Filter out points outside the boundaries
filtered_df = samp_with_coords[
    (samp_with_coords['latitude'] <= north) &
    (samp_with_coords['latitude'] >= south) &
    (samp_with_coords['longitude'] >= west) &
    (samp_with_coords['longitude'] <= east)
].copy()

# Define color mapping based on the score
def score_to_color(score):
    normalized_score = (score - 70) / (100 - 70)  # Assuming scores range from␣
 ↪70 to 100
```

```
    return plt.cm.RdYlGn(normalized_score)

# Apply color mapping
filtered_df.loc[:, 'color'] = filtered_df['score'].apply(score_to_color)

# Set the figure size
plt.figure(figsize=(12, 12))  # Adjust size as needed

# Display the image
plt.imshow(image)
plt.axis('off')  # Hide axes

# Plot the points on the image using pixel coordinates, color by score, and add␣
 ↪a very thin black outline
plt.scatter(filtered_df['x_pixel'], filtered_df['y_pixel'],
            c=filtered_df['color'], edgecolor='black', s=70, linewidth=0.4,␣
 ↪alpha=0.9)  # Adjust 's' for marker size and 'linewidth' for outline␣
 ↪thickness

plt.show()
```



We can see that vast majority of the points are colored green for an 'A' rating. Howver there are some visible yellow and red inspections on the map. Maybe we can find a pattern of some sort here.

Though we cannot at the moment convert all of the points to coordinates, we can group our inspections by the zip codes in which they took place. It will be useful to convert our `facility_zip` column to a more consistent 5 digit zip code column.

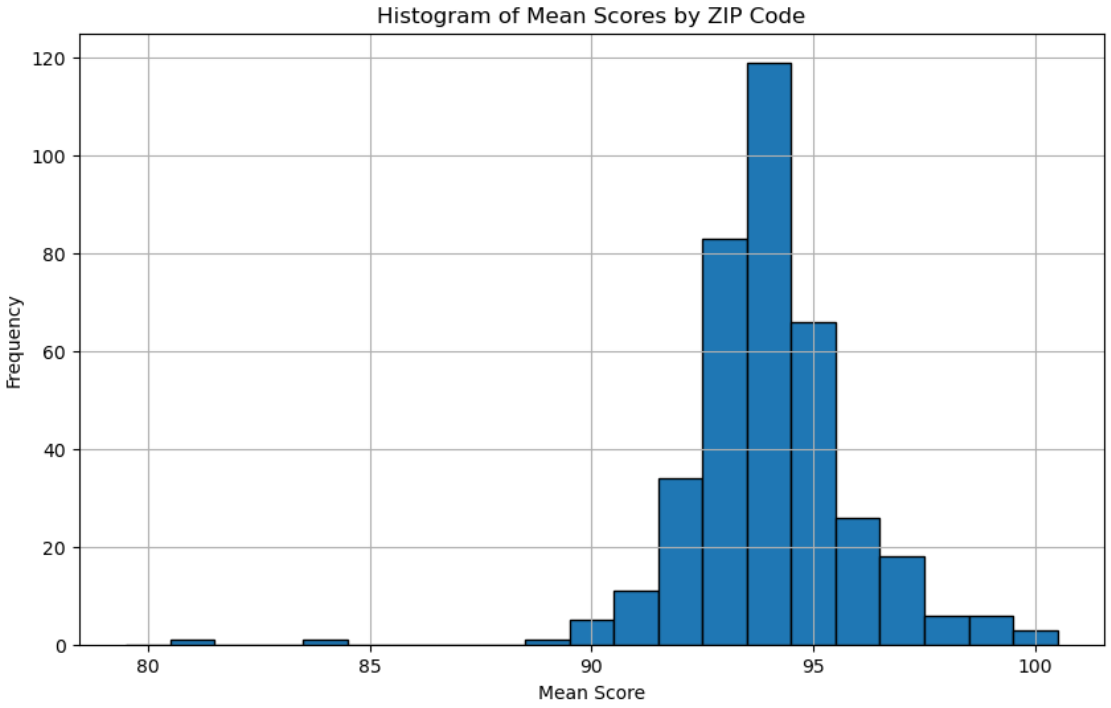
[182]:
```
# Make a new column to reduce all 5 and 9 digit 'facility_zip' codes to 5 digit␣
↪codes
inspections_df['zip_five'] = inspections_df['facility_zip'].astype(str).str.
↪slice(0, 5)
```

```
# Find unique counts
unique_zip_codes = inspections_df['zip_five'].nunique()
print(f'Inspections took place across {unique_zip_codes} zip codes in LA County.
  ↪')
```

```
Inspections took place across 380 zip codes in LA County.
```

We can visualize average inspection score by zip code

[183]:
```
# Calculate the mean score for each ZIP code
mean_scores = inspections_df.groupby('zip_five')['score'].mean().reset_index()

# Rename the columns for clarity
mean_scores.columns = ['zip_five', 'mean_score']

# Extract the 'mean_score' column for histogram plotting
scores = mean_scores['mean_score']

# Define the bins explicitly using np.arange
bin_edges = np.arange(scores.min() - 0.5, scores.max() + 1.5, 1.0)

# Define the bins explicitly to center over the values
bin_edges = np.arange(int(scores.min()) - 0.5, int(scores.max()) + 1.5, 1)

# Plot the histogram
plt.figure(figsize=(10, 6))
plt.hist(scores, bins=bin_edges, edgecolor='black')  # Adjust bins as needed
plt.title('Histogram of Mean Scores by ZIP Code')
plt.xlabel('Mean Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



From the plot we can see that there are at least two zip codes that have a noticibly low average inspection score. Lets see what these are.

[184]:
```
mean_scores.sort_values(by='mean_score', ascending=True).head()
```

[184]:

	zip_five	mean_score
322	91749	80.500000
49	90052	84.166667
264	91371	89.363636
140	90510	89.600000
326	91756	89.800000

It appears that these points come from the zip codes 91749 and 90052. Perhaps these zipcodes are areas of potential pubic health crisis.
Howver, upon inspection the two outlier zip codes contain very few inspections on only three businesses. Some internet search found that one of these zip codes is incredibly small and the other is an industrial area. We can see there are only three restaurants in these two zip codes combined.

[185]:
```
# From the two outlier zip-codes above, find all restaurant names:
inspections_df[(inspections_df['zip_five'] == str(91749)) |␣
 ↪(inspections_df['zip_five'] == str(90052))].facility_name.unique()
```

[185]:
```
array(['M.A.D COOKIN', "WU'S KITCHEN", "MOMO'S KITCHEN INC."],
      dtype=object)
```

Perhaps however if we visualize average inspection score by zipcode on a map of LA County areas of lower or higher food safety will be visible.

Visualize average inspection score by zip code

[186]:
```
# Calculate the average score for each ZIP code
zip_scores = inspections_df.groupby('zip_five')['score'].mean().reset_index()
zip_scores.columns = ['zip_five', 'mean_score']

# Load the GeoJSON file for LA ZIP codes
zip_geo = 'LA_County_Zip_Codes.geojson'
gdf = gpd.read_file(zip_geo)

# Ensure the ZIP code column in the GeoDataFrame is a string
gdf['ZIPCODE'] = gdf['ZIPCODE'].astype(str)

# Merge the ZIP code scores with the GeoDataFrame
gdf = gdf.merge(zip_scores, left_on='ZIPCODE', right_on='zip_five', how='left')

# Initialize a folium map centered around Los Angeles
m = folium.Map(location=[34.0522, -118.2437], zoom_start=10)

# Add the choropleth layer
folium.Choropleth(
    geo_data=gdf,
    name='choropleth',
    data=gdf,
    columns=['ZIPCODE', 'mean_score'],
    key_on='feature.properties.ZIPCODE',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Average Score by ZIP Code'
).add_to(m)

# Add tooltips to display ZIP code and mean score
folium.features.GeoJson(
    gdf,
    name='ZIP Codes',
    tooltip=folium.features.GeoJsonTooltip(
        fields=['ZIPCODE', 'mean_score'],
        aliases=['ZIP Code', 'Average Score'],
        localize=True
    )
).add_to(m)

# Add a layer control panel to the map
folium.LayerControl().add_to(m)
```

```
# Display the map in the Jupyter notebook
display(m)
```

```
<folium.folium.Map at 0x2a143d690>
```

The dark areas above are zip codes lacking any listed inspections. Perhaps a separate municipality claims responsibility for public health inspections in these areas. Any shade of orange or yellow, even the very lightly colored areas have average scores over 90. From the plot, nothing sticks out to me as an obvious pattern to the scores of zip codes.

However, perhaps if socioeconomic data is added we will be able to see a trend in the health inspection scores. Below we will load in median household income data for the zip codes in LA County. This data was chosen from the year 2018 since it was available and coincided with the inspection data being from 2015 to Dec 2017.

[187]:
```
# read in median household income by zip code
# data from LA Almanac website acquired from US Census
income_df = pd.read_csv('income_by_zip.csv')
```

[188]:
```
# Ensure 'zip_five' in inspections_df is numeric
inspections_df['zip_five'] = inspections_df['zip_five'].astype(str).str.
 ↪extract('(\d+)').astype(int)

# Ensure 'ZIP Code' in income_df is numeric
income_df = pd.read_csv('income_by_zip.csv')
income_df['ZIP Code'] = income_df['ZIP Code'].astype(str).str.extract('(\d+)').
 ↪astype(int)

# Merge inspections_df with income_df on the ZIP Code column
merged_df = inspections_df.merge(income_df, left_on='zip_five', right_on='ZIP␣
 ↪Code', how='inner')

# Drop one of the two zip columns
merged_df.drop(columns=['ZIP Code'], inplace=True)
```

[189]:
```
# We can see the inspections now with income data added.
# Show fields to visualize addition of Income data
columns_to_keep = ['facility_name', 'score', 'zip_five', 'Income']

# Create a subset of merged_df with only the specified columns
subset_df = merged_df[columns_to_keep]

# Display the subset DataFrame
subset_df.head()
```

[189]:

	facility_name	score	zip_five	Income
0	HOVIK'S FAMOUS MEAT & DELI	98	91344	96162.0

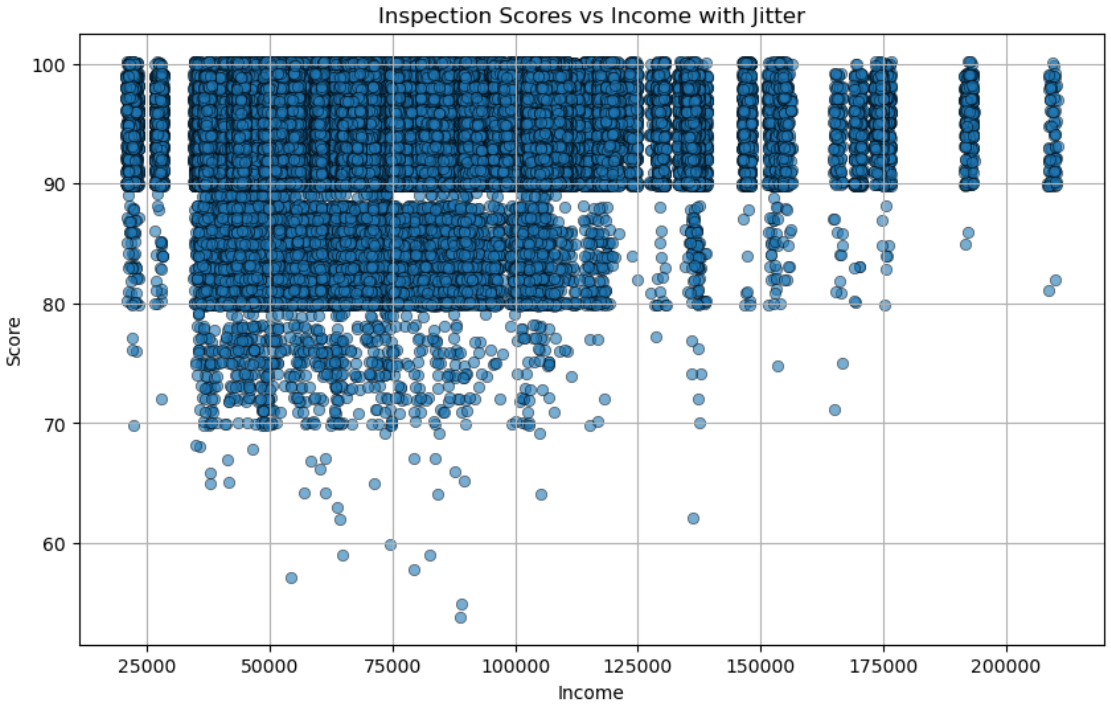
```
1              BAITH AL HALAL     95     91344  96162.0
2                  TERIYAKIYA     96     91344  96162.0
3               PIZZA COOKERY     96     91344  96162.0
4           MAIN KITCHEN CAFE     97     91344  96162.0
```

[190]:
```
# A small amount of rows were lost in the join
print(inspections_df.shape)
print(merged_df.shape)
```

```
(191371, 21)
(189630, 22)
```

Let's see how the average inspection scores per zip code look when plotted against median household income for those zip codes.

[191]:
```
# Add jitter to the x and y coordinates
x_jitter = 1000  # Adjust as needed for more x-axis jitter
y_jitter = 0.2  # Adjust as needed for more y-axis jitter

# Adding jitter to the coordinates
merged_df['score_jittered'] = merged_df['score'] + np.random.uniform(-y_jitter,␣
 ↪y_jitter, merged_df.shape[0])
merged_df['income_jittered'] = merged_df['Income'] + np.random.
 ↪uniform(-x_jitter, x_jitter, merged_df.shape[0])

# Plot the scatter plot with jittered coordinates
plt.figure(figsize=(10, 6))
plt.scatter(merged_df['income_jittered'], merged_df['score_jittered'], alpha=0.
 ↪6, edgecolor='k', linewidth=0.5)
plt.title('Inspection Scores vs Income with Jitter')
plt.xlabel('Income')
plt.ylabel('Score')
plt.grid(True)
plt.show()
```



We can see in the above plot that lower inspections scores appear to correlate with areas of lower median household income. However, there are many more zip codes of average to low income than there are of high income. We should test this to see if it is true.

[192]:
```
# Prepare the data for regression
X = merged_df['Income']
y = merged_df['score']

# Add a constant to the predictor variable set
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print out the regression results
print(model.summary())
```

OLS Regression Results

Dep. Variable:	score	R-squared:	0.006
Model:	OLS	Adj. R-squared:	0.006
Method:	Least Squares	F-statistic:	1051.
Date:	Sun, 04 Aug 2024	Prob (F-statistic):	7.02e-230
Time:	22:36:10	Log-Likelihood:	-5.2393e+05
No. Observations:	189630	AIC:	1.048e+06

```
Df Residuals:                   189628   BIC:                         1.048e+06
Df Model:                            1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         92.9741      0.025   3695.296      0.000      92.925      93.023
Income      1.055e-05   3.25e-07     32.416      0.000    9.91e-06    1.12e-05
==============================================================================
Omnibus:                    48512.852   Durbin-Watson:                   1.867
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           163382.210
Skew:                          -1.285   Prob(JB):                         0.00
Kurtosis:                       6.752   Cond. No.                     2.21e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.21e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From our regression above we can see there is a statistically significant relationship between median household income and inspections score (p ~ 0). However, based on the 'Income' coefficient of 1e-5 we can see that while there is a positive relationship it is very weak. Our R-squared value of .006 tells us that median household income explains very little of the variability in the inspection scores.

So if we wanted to use our findings here to limit our dining to areas with safer health inspection ratings, from our above analysis we would not have much cause to avoid any areas or assume that food safety is poorer in any specific areas. It seems so far that health quality, as defined by LA County Public Health, is fairly high across all geographic areas and across areas of high/med/low income.

However, if we were primarily concerned about a specific few health code violations we could perhaps find where those are more common.

1.2.1 The critical violations inspection list:

We can choose a few violations to prioritize:

11 point violations: *recent law has given these violations increased point deduction values
21b. Water available
22. Sewage and wastewater properly disposed
23. No insect, rodent, birds or animals present

As these are the most significant violations, we will examine the rates and such violations and where they occur.

[193]:
```
# Adding a new field, 'violation_number' because the 'violation_code' values␣
↪can be confusing.
# Function to extract the violation number
```

```
def extract_violation_number(description):
    match = re.match(r'#\s*([0-9]{1,2}[a-zA-Z]?)', description)
    return match.group(1) if match else None

# Apply the function to create a new column
violations_df['violation_number'] = violations_df['violation_description'].
 ↪apply(extract_violation_number)

# Function to search for a restaurant name and specified violation
def search_violation(restaurant_name, violation_number):
    filtered_df = violations_df[
        (violations_df['restaurant_name'].str.contains(restaurant_name,␣
 ↪case=False)) &
        (violations_df['violation_number'] == violation_number)
    ]
    return filtered_df

violations_df.head()
```

[193]:

	points	serial_number	violation_code	violation_description	violation_status	violation_number
0	1	DAJ5UNMSF	F044	# 44. Floors, walls and ceilings: properly bui…	OUT OF COMPLIANCE	44
1	4	DAT2HKIRE	F007	# 07. Proper hot and cold holding temperatures	OUT OF COMPLIANCE	07
2	1	DAT2HKIRE	F033	# 33. Nonfood-contact surfaces clean and in go…	OUT OF COMPLIANCE	33
3	1	DAT2HKIRE	F035	# 35. Equipment/Utensils - approved; installed…	OUT OF COMPLIANCE	35
4	1	DAQN0I8EA	F033	# 33. Nonfood-contact surfaces clean and in go…	OUT OF COMPLIANCE	33

[194]:
```
# Combine violations with restaurant name and id
violations_merged = violations_df.merge(inspections_df[['serial_number',␣
 ↪'facility_name', 'facility_id']], on='serial_number', how='left')
# dropping where facility name is n/a removes about 1000 rows
violations_merged = violations_merged.dropna(subset=['facility_name',␣
 ↪'facility_id'])
violations_merged.head(2)
```

[194]:
```
   points serial_number violation_code  \
0       1     DAJ5UNMSF           F044
1       4     DAT2HKIRE           F007

                               violation_description   violation_status  \
0  # 44. Floors, walls and ceilings: properly bui…  OUT OF COMPLIANCE
1     # 07. Proper hot and cold holding temperatures  OUT OF COMPLIANCE

  violation_number      facility_name facility_id
0               44  ALTA DENA EXPRESS   FA0003460
1               07  ANGKOR RESTAURANT   FA0159686
```

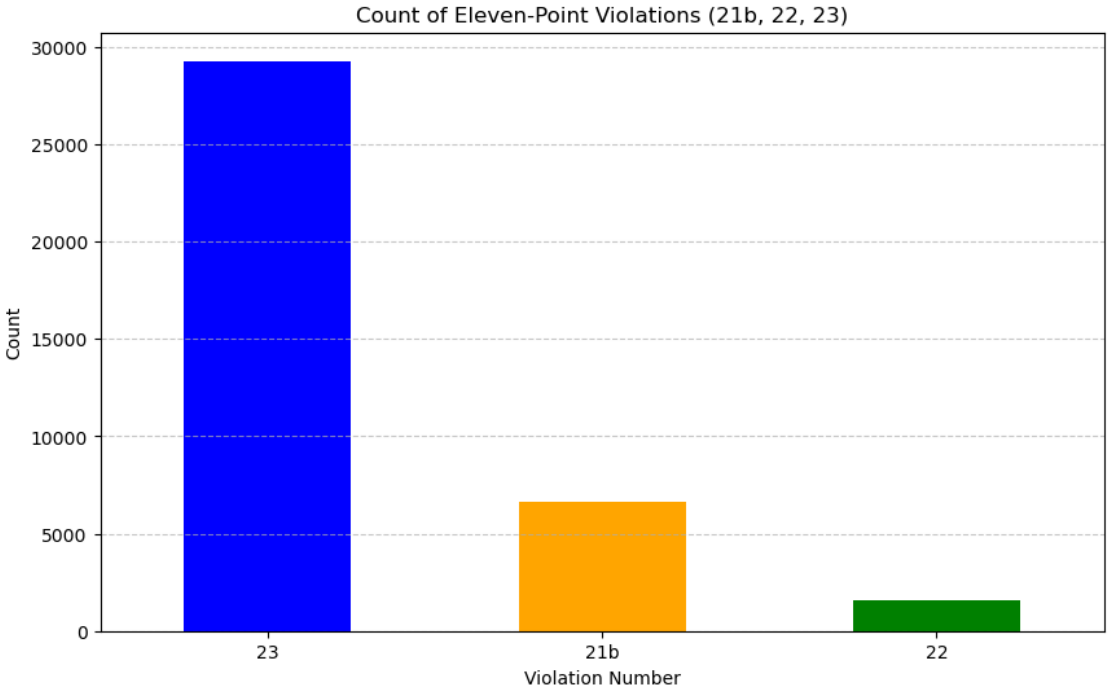
We can see how many of these 11 point violations are contained in violations_df:

[195]:
```
# Make a set of the most extreme violation numbers:
eleven_point_violations = {'21b', '22', '23'}

# Filter for eleven-point violations
eleven_point_violations_df =␣
 ↪violations_merged[violations_merged['violation_number'].
 ↪isin(eleven_point_violations)]

eleven_point_violations_df.shape
```

[195]:
```
(37504, 8)
```

Let's look at the frequency of these three violations:.

[196]:
```
# Count the occurrences of each violation number in the eleven-point violations
violation_counts = eleven_point_violations_df['violation_number'].value_counts()

# Plot the histogram
plt.figure(figsize=(10, 6))
violation_counts.plot(kind='bar', color=['blue', 'orange', 'green'])
plt.title('Count of Eleven-Point Violations (21b, 22, 23)')
plt.xlabel('Violation Number')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



We can see that the most common violation here is #23 which relates to the presence of insects, rodents, or birds in the restaurant. Let's visualize the areas where these occur. Below we will plot the number of such violations that occurred per zip code divided by the total number of inspections in a given zip code.

[197]:
```
# Merge eleven_point_violations_df with inspections_df to get the ZIP codes
merged_df_violations_zip = pd.merge(
    eleven_point_violations_df,
    inspections_df[['facility_id', 'zip_five']],
    on='facility_id',
    how='left'
)

# Remove duplicates to ensure only unique violation records are counted
merged_df_violations_zip = merged_df_violations_zip.
 ↪drop_duplicates(subset=['serial_number', 'facility_id', 'violation_number',␣
 ↪'zip_five'])

# Ensure 'zip_five' column in merged_df_violations_zip is string
merged_df_violations_zip['zip_five'] = merged_df_violations_zip['zip_five'].
 ↪astype(str)

# Count the number of eleven-point violations per ZIP code
```

```
violation_counts_by_zip = merged_df_violations_zip.groupby('zip_five').size().
 ↪reset_index(name='violation_count')

# Calculate the total number of inspections per ZIP code
total_inspections_per_zip = inspections_df.groupby('zip_five').size().
 ↪reset_index(name='total_inspections')

# Ensure 'zip_five' column in total_inspections_per_zip is string
total_inspections_per_zip['zip_five'] = total_inspections_per_zip['zip_five'].
 ↪astype(str)

# Merge the total inspections with the violation counts
violation_counts_by_zip = violation_counts_by_zip.
 ↪merge(total_inspections_per_zip, on='zip_five', how='left')

# Calculate the normalized violation count
violation_counts_by_zip['normalized_violation_count'] =␣
 ↪violation_counts_by_zip['violation_count'] /␣
 ↪violation_counts_by_zip['total_inspections']

# Load the GeoJSON file for LA ZIP codes
zip_geo = 'LA_County_Zip_Codes.geojson'
gdf = gpd.read_file(zip_geo)

# Ensure the ZIP code column in the GeoDataFrame is a string
gdf['ZIPCODE'] = gdf['ZIPCODE'].astype(str)

# Merge the normalized violation counts with the GeoDataFrame
gdf = gdf.merge(violation_counts_by_zip, left_on='ZIPCODE',␣
 ↪right_on='zip_five', how='left')

# Initialize a folium map centered around Los Angeles
m = folium.Map(location=[34.0522, -118.2437], zoom_start=10)

# Add the choropleth layer
folium.Choropleth(
    geo_data=gdf,
    name='choropleth',
    data=gdf,
    columns=['ZIPCODE', 'normalized_violation_count'],
    key_on='feature.properties.ZIPCODE',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Normalized Count of Eleven-Point Violations by ZIP Code'
).add_to(m)
```

```
# Add tooltips to display ZIP code and normalized violation count
folium.features.GeoJson(
    gdf,
    name='ZIP Codes',
    tooltip=folium.features.GeoJsonTooltip(
        fields=['ZIPCODE', 'normalized_violation_count'],
        aliases=['ZIP Code', 'Normalized Violation Count'],
        localize=True
    )
).add_to(m)

# Add a layer control panel to the map
folium.LayerControl().add_to(m)

# Display the map in the Jupyter notebook
display(m)
```

```
<folium.folium.Map at 0x2918d5750>
```

From the map there are clear areas of elevated violation rates. However many of these are areas of few inspections.

[198]:
```
# Check the violation counts by ZIP code
violation_counts_by_zip.sort_values(by='normalized_violation_count',␣
 ↪ascending=False).head(10)
```

[198]:

	zip_five	violation_count	total_inspections	normalized_violation_count
132	90635	1	1	1.000000
277	91749	3	4	0.750000
166	90921	2	3	0.666667
47	90052	4	6	0.666667
68	90103	3	5	0.600000
145	90704	123	237	0.518987
259	91609	1	2	0.500000
242	91416	1	2	0.500000
70	90207	1	2	0.500000
133	90637	2	4	0.500000

However it does appear that some areas have elevated rates of these major infractions. This information could be useful in the distribution of public health advertisement and funding. Individuals frequenting or living in these areas could be interested in pushing for safer restaurants. Though the rates are not much higher than average.

[199]:
```
# Average normalied violation count
violation_counts_by_zip['normalized_violation_count'].mean()
```

[199]:
```
0.212608901157589
```

[200]:
```
# Filter for total_inspections > 100 and sort by 'normalized_violation_count'␣
 ↪in descending order
filtered_sorted_violation_counts_by_zip =␣
 ↪violation_counts_by_zip[violation_counts_by_zip['total_inspections'] > 100].
 ↪sort_values(by='normalized_violation_count', ascending=False)

# Display the first 20 rows of the filtered and sorted DataFrame
filtered_sorted_violation_counts_by_zip.head(10)
```

[200]:

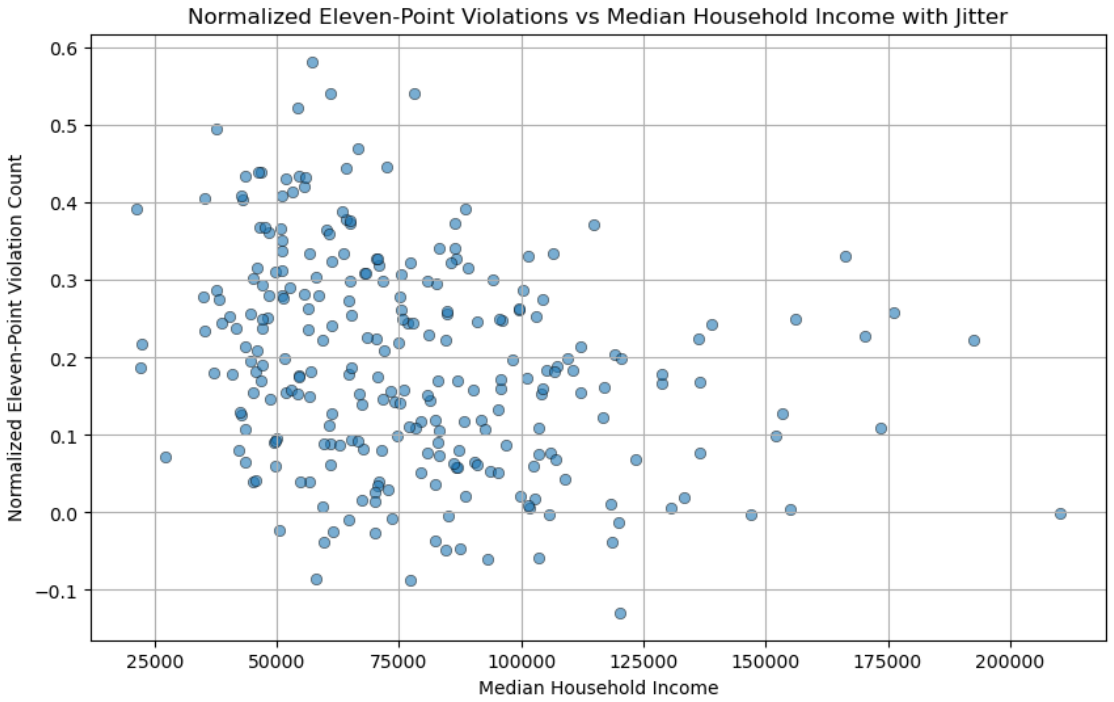
	zip_five	violation_count	total_inspections	normalized_violation_count
145	90704	123	237	0.518987
55	90063	313	724	0.432320
2	90003	383	910	0.420879
222	91352	254	606	0.419142
41	90044	239	618	0.386731
310	93543	44	114	0.385965
280	91755	86	235	0.365957
72	90211	136	372	0.365591
9	90011	406	1132	0.358657
4	90005	476	1343	0.354430

We can see that though elevated, these ares still have relatively few total inspections and have rates fewer than 3 times the average.

We can see that though elevated, these ares still have relatively few total inspections and have rates fewer than 3 times the average.
Lets again compair to median household income. Plot:

[201]:
```
# Ensure the columns are of integer type before merging
violation_counts_by_zip['zip_five'] = violation_counts_by_zip['zip_five'].
 ↪astype(int)
income_df['ZIP Code'] = income_df['ZIP Code'].astype(int)

# Merge income data with violation_counts_by_zip on the ZIP code
violation_counts_by_zip = violation_counts_by_zip.merge(income_df,␣
 ↪left_on='zip_five', right_on='ZIP Code', how='left')

# Drop the redundant 'ZIP Code' column
violation_counts_by_zip.drop(columns=['ZIP Code'], inplace=True)

# Add jitter to the x and y coordinates
x_jitter = 1000  # Adjust as needed for more x-axis jitter
y_jitter = 0.2  # Adjust as needed for more y-axis jitter

# Adding jitter to the coordinates
violation_counts_by_zip['normalized_violation_count_jittered'] =␣
 ↪violation_counts_by_zip['normalized_violation_count'] + np.random.
 ↪uniform(-y_jitter, y_jitter, violation_counts_by_zip.shape[0])
```

```
violation_counts_by_zip['income_jittered'] = violation_counts_by_zip['Income']␣
↪+ np.random.uniform(-x_jitter, x_jitter, violation_counts_by_zip.shape[0])

# Plot the scatter plot with jittered coordinates
plt.figure(figsize=(10, 6))
plt.scatter(violation_counts_by_zip['income_jittered'],␣
↪violation_counts_by_zip['normalized_violation_count_jittered'], alpha=0.6,␣
↪edgecolor='k', linewidth=0.5)
plt.title('Normalized Eleven-Point Violations vs Median Household Income with␣
↪Jitter')
plt.xlabel('Median Household Income')
plt.ylabel('Normalized Eleven-Point Violation Count')
plt.grid(True)
plt.show()
```



Like before, we can potentially see some trend in this data, though points are sparser at higher average incomes.

[202]:
```
# Remove rows with NaNs or infinite values in 'Income' or␣
↪'normalized_violation_count'
violation_counts_by_zip_clean = violation_counts_by_zip.replace([np.inf, -np.
↪inf], np.nan).dropna(subset=['Income', 'normalized_violation_count'])

# Prepare the data for regression
```

```python
X = violation_counts_by_zip_clean['Income']
y = violation_counts_by_zip_clean['normalized_violation_count']

# Add a constant to the predictor variable set
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print the summary of the regression
print(model.summary())
```

```
                                OLS Regression Results
==============================================================================
======
Dep. Variable:     normalized_violation_count   R-squared:
0.252
Model:                                   OLS   Adj. R-squared:
0.249
Method:                        Least Squares   F-statistic:
88.42
Date:                       Sun, 04 Aug 2024   Prob (F-statistic):
2.72e-18
Time:                               22:36:16   Log-Likelihood:
329.90
No. Observations:                        265   AIC:
-655.8
Df Residuals:                            263   BIC:
-648.6
Df Model:                                  1
Covariance Type:                   nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2973      0.012     25.786      0.000       0.275       0.320
Income       -1.3e-06   1.38e-07     -9.403      0.000   -1.57e-06   -1.03e-06
==============================================================================
Omnibus:                       26.682   Durbin-Watson:                   1.518
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               35.346
Skew:                           0.694   Prob(JB):                     2.11e-08
Kurtosis:                       4.129   Cond. No.                     2.24e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.24e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

We see from the regression that there is a significant relationship between median household income per zip code and normalized violation count per zip code. There is a weak negative relationship. As household income increases in an area, the rates of major violations decreases slighly.

1.3 Discussion

Overall we did not find anything all that interesting. We did find that inspection scores are fairly high across all of LA County, more or less regardless of the wealth of an area. In poorer areas restaurants are more likey to have serious violations, but these rates seem fairly even. I think that their could potentially be more visible difference in food safety if we looked at type of food and or style of restaurant. Such as Mexican, Chinese, Burgers etc or Deli, Buffet, Diner etc. Since this is dependent on the quality and characteristics of specifically LA County inspections, it could be beneficial to bring in inspection data from other locations as well as from sources other than regulatory bodies.

[ ]:

[ ]: