MSBA-320 Final Project

2022 SF Street & Sidewalk Report Replication and Regression vs Neighborhood Ratings

Nathan Torento

MSBA, Golden Gate University

MSBA 320: Advanced Statistical Analysis with R and Python

Professor Siamak Zadeh

August 17, 2023

Table of Contents

Introduction

In 2022, the SF Controller's Office released the "2022 Street & Sidewalk Maintenance Standards Report" online on the sf.gov website (SF Controller's Office, 2023a). The report analyzes its own dataset to measure the cleanliness and appearance of public streets and sidewalks according to the presence of different kinds of litter observed. It would actually be more accurate to refer to each recorded variable as an 'uncleanliness feature'. For those that have lived in SF, the results aren't shocking. Different neighborhoods display different levels of uncleanliness. The report mainly features multiple scores as percentages. When someone randomly visits any street segment in the city, this is the likelihood they would observe a certain level of the uncleanliness feature. As a San Francisco resident passionate about using data for good, I thought it would be a rewarding learning experience to recreate the results.

On the other hand, as an SF resident, I'm also constantly thinking about which neighborhoods are best to live in and how I can improve them. The SF government also maintains a ranking of all neighborhoods according to different factors like "Muni and Transportation" or "Safety and Policing". A rating that is particularly relevant is the "2023 Street and sidewalk ratings by neighborhood" that uses a separate survey dataset (SF Controller's Office, 2023c). So for my final project, I wanted to use these two datasets to achieve two objectives: 1) to replicate all the main results of the "2022 Street & Sidewalk Maintenance Standards Report", and 2) to understand which of the uncleanliness features are likely to affect a neighborhood's rating. Objective 1 will require a descriptive analysis of the dataset, while Objective 2 will require a correlation analysis and regression analysis.

Data Collection

Data Collection: Survey Results Descriptive Analysis and Replication

The dataset used by the report is based on 3,000+ manual evaluations conducted by an evaluation team of streets and sidewalks across San Francisco across 14 uncleanliness features (SF Controller's Office, 2023e). In reality, there are multiple columns that test out a different aspect of each feature, and there are other non-feature-related columns such as 'ObjectID', 'Route Location', 'x', 'y', and 'CreationDate'. However, only 14 of those columns represent these feature columns used by the report for their results. Their column names are the respective question being asked in order to fill its value. The report splits up the dataset into two categories that they call the Core Citywide Survey and the special Key Commercial Areas Survey. The Core Citywide Survey is a random sample of 1000, with 80% of the sample being residential streets and sidewalks. The Key Commercial Areas Survey is a sample of 769 high-use corridors sampled from

commercial, industrial, or mixed-use street segments. In the dataset, there's a column "Is this route predominantly residential or commercial/industrial/mixed use?" where 1=residential and 2=commercial/industrial/mixed. For this analysis, I was able to recreate the criteria for the former using a random sample where 80% have a value of 1 and 20% have a value of 2 for this column. In the case of recreating the Key Commercial Areas, it was unclear what made a data point a "high-use" corridor, nor were there any instructions on the metadata or the report on how to identify them. Thus, I simply obtained a random sample of 769 rows where they all have a value of 1 for the relevant column.

The rows in the dataset identify different street 'features' such as litter, sidewalk litter, sidewalk pavement condition, illegal dumping, hazards (and type of hazard ex: feces, syringe), graffiti, transit shelters. There are 10 main findings which we will attempt to replicate. "64% of key commercial area evaluations have street litter". "44% of evaluations have sidewalk litter". "8% of evaluated sidewalks had overflowing trash". "More than 30% of evaluated sidewalks had clearance issues". "75% of evaluated sidewalks have moderate to severe pavement defects". "More than 30% of evaluations report illegal dumping". "More than 80% of transit shelters on evaluated streets & sidewalks had cleanliness issues". "About 20% of evaluated streets & sidewalks have graffiti, but graffiti more common in commercial areas". "Almost 50% of city streets and sidewalks report broken glass". "About 30% of evaluated streets and sidewalks report feces". Table 1 below shows a summarized version of the results, along with percentages for the other remaining features.

Table 1

Results Overview of 2022 Street & Sidewalk Maintenance Standards Report

Feature	Core Citywide Survey (% issue present)	Key Commercial Areas (% issue present)
Street litter	41% moderate to severe	64% moderate to severe
Sidewalk litter	44% moderate to severe	67% moderate to severe
Overflowing trash receptacles	8%	11%
Sidewalk clearance	31%	12%
Sidewalk pavement defects	75% moderate to severe	75% moderate to severe
Illegal dumping	36%	49%
Broken glass	47%	58%
Feces	30%	47%
Syringes	1%	3%
Condoms	0.1%	0.5%
Dead animals	5%	0.5%
Odors	2%	3%
Graffiti	20% moderate to severe	71% moderate to severe
Transit shelters	83%	91%

Note. This table shows the percentage of an uncleanliness feature's prevalence for each regional category "Core Citywide Survey" and "Key Commercial Areas". Taken from https://sf.gov/reports/may-2023/2022-street-sidewalk-maintenance-standards-report#results-overview

Data Collection: Regression Analysis

As part of an overall City Survey whose results are available online as a dataset, residents are asked to rate their perception of the cleanliness and condition of streets and sidewalks. (SF Controller's Office, 2023b). The city conducts a weighted analysis biennially to score neighborhoods on a numeric scale of 1 to 5 and a corresponding letter grade, across different factors, as well as an overall score. There is thorough documentation on the methodology behind the methodology of the weighted analysis (SF Controller's Office, 2023b). I will not be attempting to replicate the weighted analysis, however. Instead, the 2nd dataset I'll be using for the expanded analysis is not the city survey's .csv file, rather it is the results of that weighted analysis as a score of the neighborhoods. It is available in view-only form on the table online online[1]. To copy the data, I manually saved the scores as a Python dictionary. In my analysis, in the 1st dataset, I create a new column called 'neighborhood_scores', then use that dictionary to map out the scores for every row according to their neighborhood. I then drop the previous neighborhood column, this way the 1st dataset now has only this one numerical predictor variable. Notice that the neighborhood ratings are for 2023 and that the dataset used to calculate these values are completely different from the dataset used by the 2022 survey results. It is out of my own personal interest and a lack of access to 2022 neighborhood rating results that I connect the two.

Methodology

Methodology: Survey Results Descriptive Analysis and Replication

To recreate the survey results and essentially conduct a descriptive analysis of the dataset, I used Python and documented all my steps onto a Jupyter Notebook. My first step was to explore and clean the dataset mainly by fixing the column data types and filling in missing values with the mean for each column. The values of the variables mostly range from 0 to 5, we are working with such a small and similar range of values overall. We also randomly sampled from the overall dataset, which is also a random sample of the city, so we can assume that the values for all columns follow some normal distribution, though we can test for this later. Assuming no skew and a normal distribution, I cleaned the data by filling in the empty values with the average value for each column. Then, I recreated the "Core Citywide Survey" and "Key Commercial Areas

[1]See online at https://sf.gov/data/city-survey-streets-and-sidewalks

Survey" data subsets/categories. Outside of this notebook, I saved the results overview table in Table 1 in the "Results Overview" section of the report onto a .csv file. I then loaded that dataset into a variable results_overview. I created two extra columns for both categories appended with '(Replicated)' to store the calculated percentages later.

The most important step for this replication is me creating this custom function replicate_survey_results(). It takes in a column name of interest, after which it does a few things: it calculates the percentages for the data subsets, it visualizes a histogram distribution of the results, then it fills in the empty (Replicated) column values accordingly. The replicate_survey_results() function mainly works by identifying the criteria according to how it's described in the report, then getting that percentage over the entire sample. To know the correct criteria for every variable, I referred to the instructions in the metadata of the dataset (SF Controller's Office, 2023e). For example, for a variable and question like "Select the statement that best describes the amount and distribution of litter in the street.", there are 5 possible values from 1-5, where 1 = None, 2 = A few traces, 3 = More than a few traces, but no accumulation, 4 = Distributed litter with some accumulation, and 5 = Widespread litter with significant accumulation. The criteria says that "moderate to severe" levels are 3-5 so that is what my function calculates. I then apply the function to recreate every main result in the report and comment on the similarities or differences of my findings.

Methodology: Regression Analysis

In the second half of the notebook, I document the steps I took to understand which of the uncleanliness features are likely to affect a neighborhood's rating through a correlation analysis and regression analysis. The correlation analysis is really only a supporting step to the main regression analysis that aims to understand which of the variables are most important in predicting the neighborhood's rating. To do that, I isolated only all the relevant predictor variables onto a new table, excluding columns like 'ObjectID' or 'CreationDate'. Note that although there are only 14 columns that represent the uncleanliness features highlighted by the report, I kept the other relevant feature-related variables as they also provide insight into the place's appearance of cleanliness. For example, there is a column specifying the count of large abandoned items in the area. In contrast, I then excluded columns that don't have clear instructions on the metadata nor on the report on how to interpret their values. For example, the column 'If yes, identify the cause (select all that apply)' was removed because its description only says 'Sidewalk clearance. Codes' without explanation as to what the codes mean anywhere. Afterwards, I created a new column 'neighborhood_street_and_sidewalk_rating' which is our response variable of the neighborhood rating. I then create a correlation matrix excluding the y variable so as to test for autocorrelation and potentially know which variables to remove for later.

In the end, I then created several multiple linear regression using the OLS function from the statsmodels.api package. I created 5 multiple linear regression models. For each model, I then created scattered plots with a line of best fit as well as diagnostic Q-Q and residual plots. Model 1 contains all predictor variables. Model 2 contains all predictor variables with p-value less than 0.05 according to Model1. Model 3 contains predictor variables that I cut according to correlation plot results. Model 4 contains predictor variables with p-value less than 0.05 according to Model3. Model 5 calculates the variables by performing a Backwards Stepwise Regression. Although there is controversy with how the approach overfits and creates a false confidence in the model, it is relatively reliable for a smaller amount of explanatory variables, simple to implement, and can still provide great insight (Smith, 2018). To assess which of the models was best (and which variables are the best predictors), I compared their Adjusted R-squared by value and through a bar chart comparison.

Results of Data Analysis

Results of Data Analysis: Survey Results Descriptive Analysis and Replication

Figure 1

Replication code example: '44% of evaluations have sidewalk litter result'

44% OF EVALUATIONS HAVE SIDEWALK LITTER

"Similar to street litter, 44% of the sampled sidewalks had moderate to severe levels of sidewalk litter in the Core Citywide Survey. However, there was much more sidewalk litter in Key Commercial Areas at 67%."

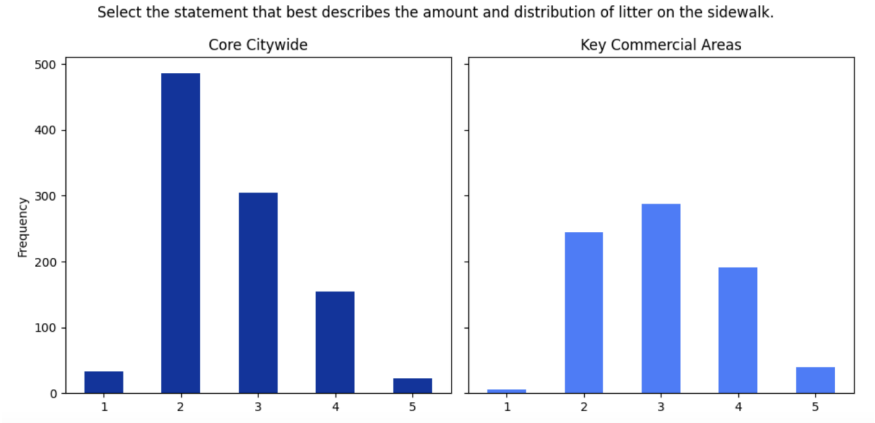
- sidewalk_litter_dist: Select the statement that best describes the amount and distribution of litter on the sidewalk.
  - Values: Sidewalk litter rated on a scale of 1-5.
    - 1 = None, 2 = A few traces, 3 = More than a few traces, but no accumulation, 4 = Distributed litter with some accumulation, 5 = Widespread litter with significant accumulation.

PARTIAL-SUCCESS: We found 48% sidewalk litter in Key Commercial Areas, which is close to the claimed 44%.
SUCCESS: We found 67% sidewalk litter in Core Citywide Survey as opposed to the claimed 44%.

```
# Create variable to represent the long column name written in the form of a question
sidewalk_litter = 'Select the statement that best describes the amount and distribution of litter on the sidewalk.'

# Perform replication, save results onto table, and display results
results_overview.loc[1,[core, key]] = replicate_survey_results(sidewalk_litter, 'sidewalk_litter')
```

```
48.10% sidewalk litter in Core Citywide Survey
67.49% sidewalk litter in Key Commercial Areas
```



Note. This is a screenshot example of one code cell wherein I use my custom "replicate_survey_results()" to do a few things: calculate the percentage for Core Citywide and Key Commercial Areas, visualize the histogram distribution of all their possible values, then store their values onto the results_overview table. In this figure, we're replicating the result '44% of evaluations have sidewalk litter'.

Table 2

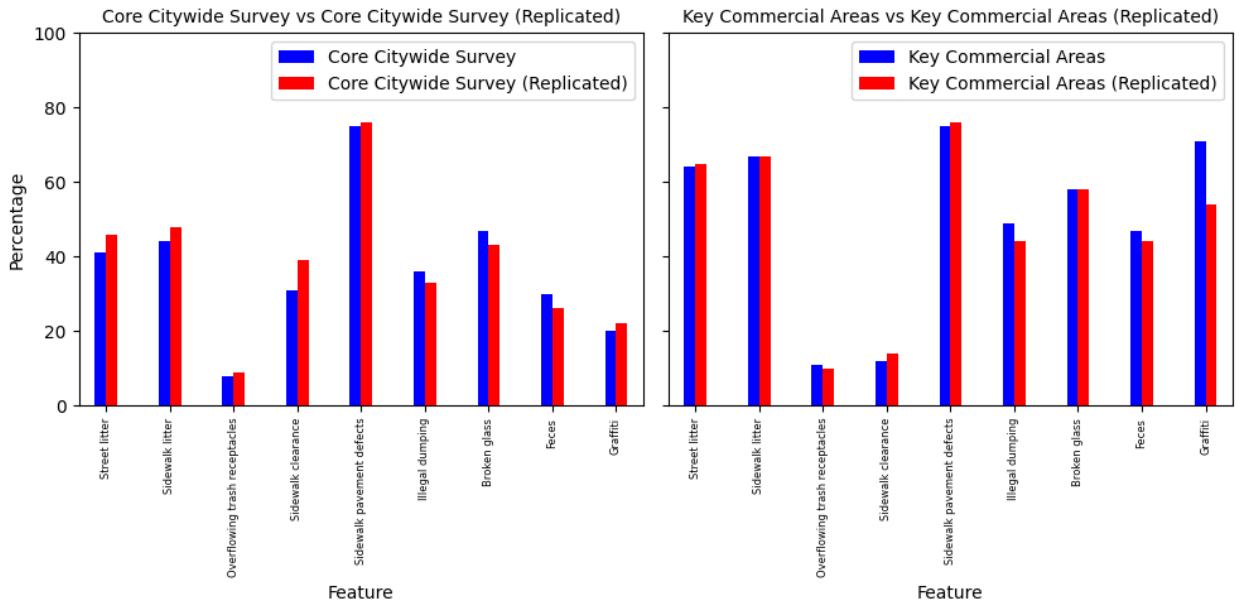
Results Overview with Replicated Results

Feature	Core Citywide Survey	Key Commercial Areas	Core Citywide Survey (Replicated)	Key Commercial Areas (Replicated)
Street litter	41.0	64.0	46.0	65.0
Sidewalk litter	44.0	67.0	48.0	67.0
Overflowing trash receptacles	8.0	11.0	9.0	10.0
Sidewalk clearance	31.0	12.0	39.0	14.0
Sidewalk pavement defects	75.0	75.0	76.0	76.0
Illegal dumping	36.0	49.0	33.0	44.0
Broken glass	47.0	58.0	43.0	58.0
Feces	30.0	47.0	26.0	44.0
Graffiti	20.0	71.0	22.0	54.0

Note. This table shows the percentage of an uncleanliness feature's prevalence for "Core Citywide Survey" and "Key Commercial Areas" and their respective "Replicated Values". Figure 2 below makes this easier to understand and digest visually.

Figure 2

'Core Citywide Survey' and 'Key Commercial Areas' prevalence by 'Feature' vs respective (Replicated) values

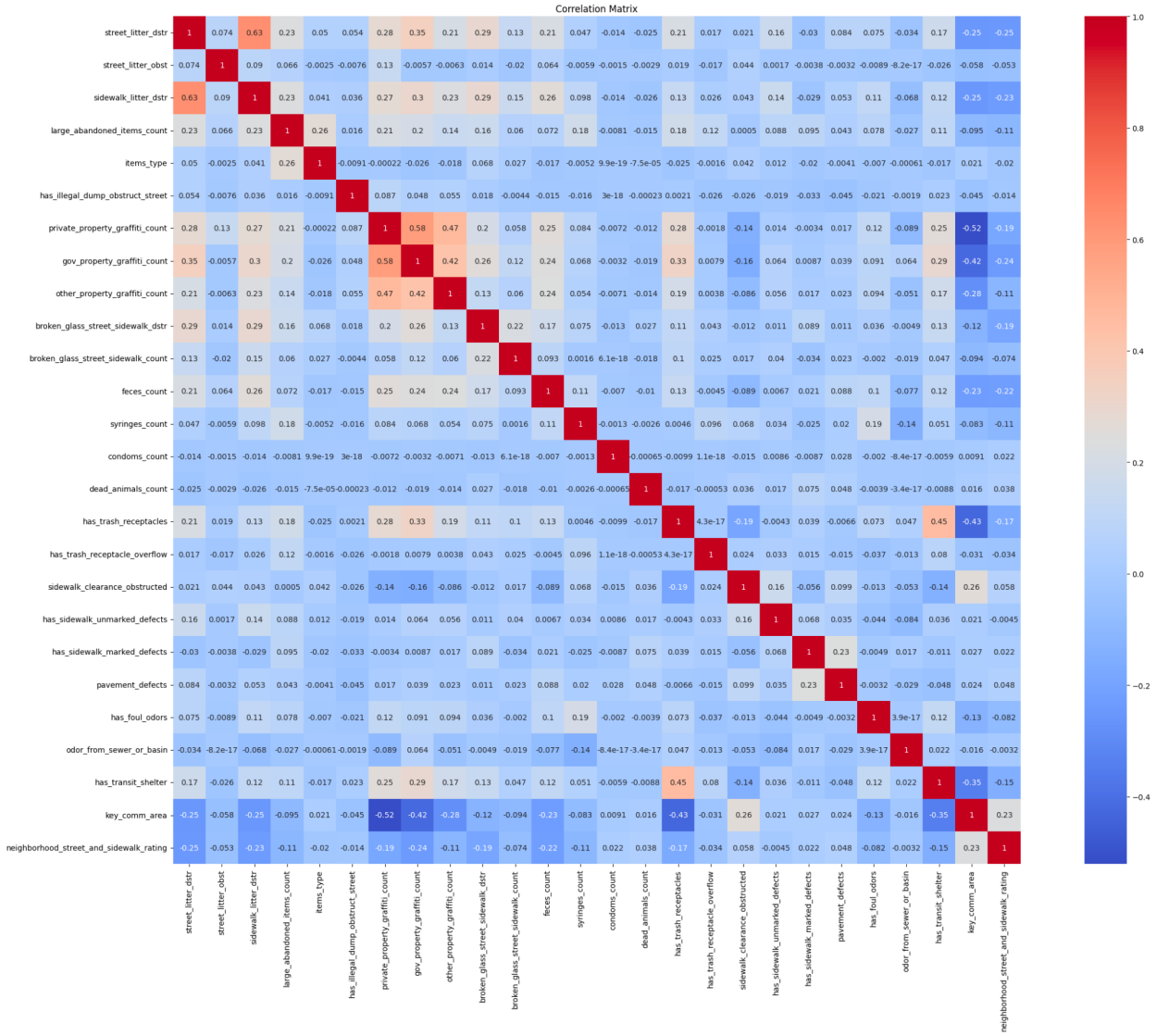


Note. There are two charts in this figure. For each chart, we have on the x-axis the name of the uncleanliness 'Features', and a shared y-axis of the percentage prevalence. In blue, we have the values for subcategory by either "Core Citywide Survey' and 'Key Commercial Areas'. In red right next to it, we have the replicated values. As we can see, although there are no glaring discrepancies, there are slight ones. For the Core Citywide Survey, the most visible differences are in 'Street litter', 'Sidewalk litter' and 'Sidewalk clearance' with a 3-5% difference. For the Key Commercial Areas, the most visible differences are in 'Illegal dumping' and 'Graffiti'.

Results of Data Analysis: Regression Analysis

Figure 3

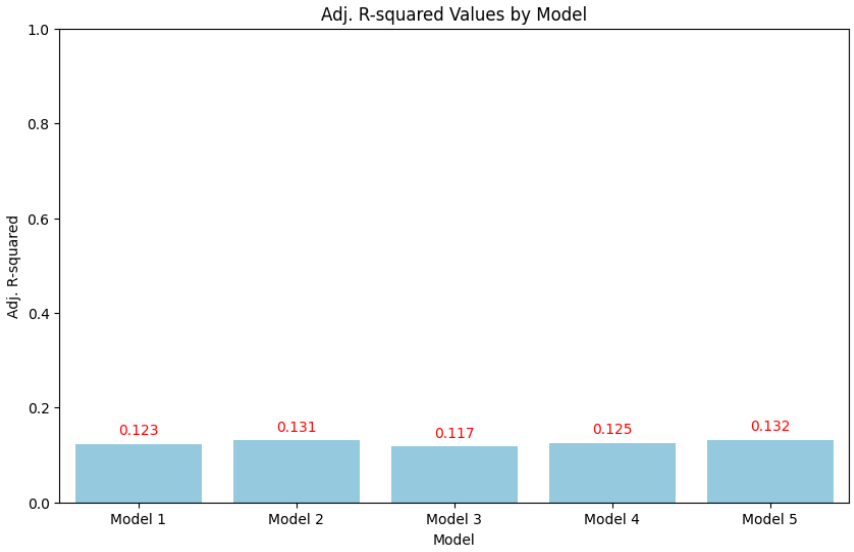
Correlation Matrix of Complete Dataset



Note. This is a correlation matrix of all the response variables that were relevant to indicating the appearance of cleanliness in the sampled area. We visualize the matrix and isolate values around or above 50% as part of our correlation analysis to determine which variables to cut to minimize autocorrelation. Most noticeably, we observe a few things. At 0.63, street_litter_dstr and street_litter_dstr are moderately positively correlated. All the graffiti variables have a moderately positive correlation around 50% to each other.

Figure 4

Adj. R-squared Values by Model



Note. A bar chart of the Adjusted R-squared values for all 5 models. Notice how these are still quite low when it comes to R-squared values. In a social sciences context, even though a 0.5 value could be considered strong, these values are within the range of 0.117 to 0.132. Although Model 5 is the strongest followed closely by Model 2, the values overall aren't too different.

Analysis and interpretation of the results

Analysis and interpretation of the results: Survey Results Descriptive Analysis and Replication

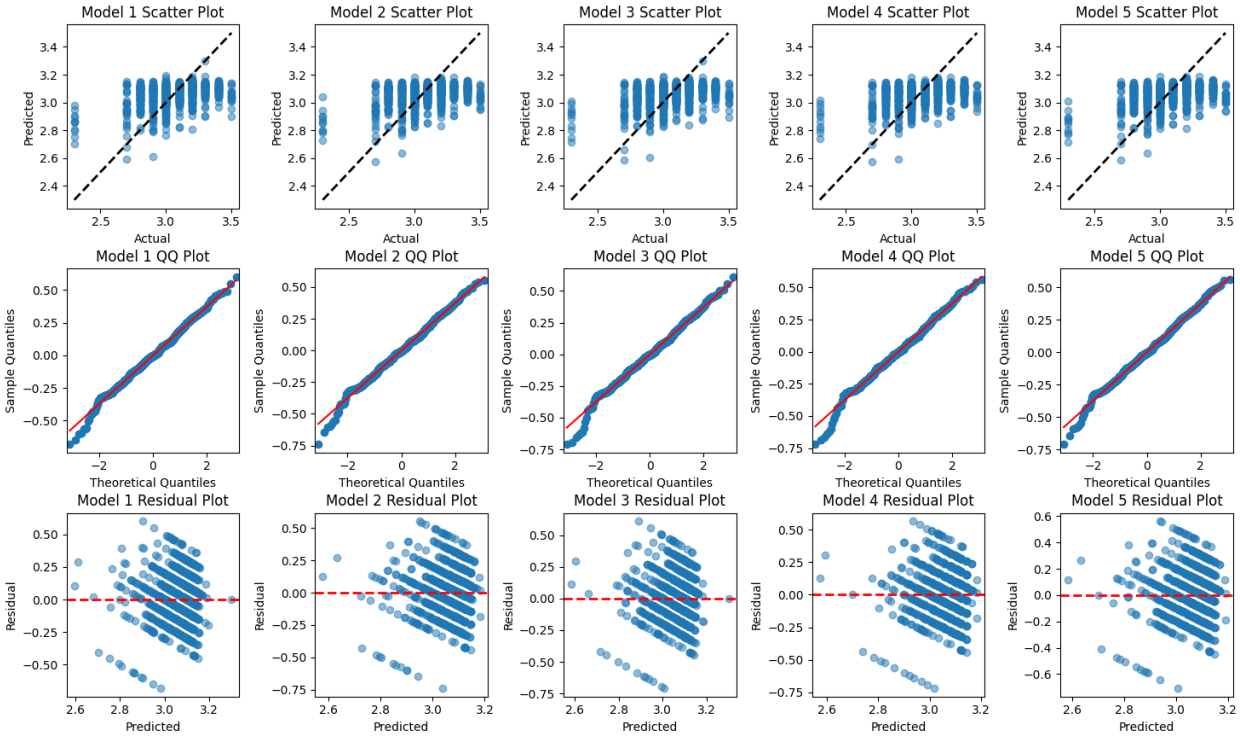
In terms of the replication, we were able to replicate 9 of the main findings. Out of 8 findings, we replicated values within 1-3% which could be considered accurate enough. In terms of the exact values, overall the differences ranged between 0 and 17%. While the rest of the values hovered between a 0-8% difference, the graffiti feature was the outlier at 17%. It is likely that this error is because the method for

which I calculated the value is incorrect. There were three graffiti questions/variables, one counting the instances of graffiti on private property, the other counting those on other property nearby, and the final one on government property. The instances are then grouped as such: None-Minor is 0-25 instances, Moderate is 26-100 instances, and Severe is more than 100 instances. The finding in the report, however, does not specify which of the three graffiti variables were used or how they were aggregated, so I assumed a proxy variable that took the max value of the three. Furthermore, there was one finding where we replicated only partially successful in that we found 48% sidewalk litter in Key Commercial Areas, which is close to the claimed 44%. The reason for these deviations is most likely due to the nature of our 'Core Citywide Survey' subset and 'Key Commercial Areas' subset being a random sample of the entire dataset. Although we performed a random sample like the report author instructed, as is inherent with random samples, there is no guarantee, and it is in fact extremely unlikely, that I have the exact same samples. This will inevitably lead to some deviation. This is inevitable in the world of statistics. Though it could be mitigated if the report author released the code they used, getting slightly different values is a good indication that the random sample is indeed representative of the population.

Analysis and interpretation of the results: Regression Analysis

Figure 5

Diagnostic Plots by Model



Note. Every column holds chart figures for each model. For each model, there are three rows of figures: a scatter plot of the actual values vs the predicted values, a Q-Q plot, and a plot of the predicted against the residuals.

The first thing we'll notice about all these plots is that they are pretty much exactly similar visually, save for one or two points if you really zoom in. This, along with the minute deviations between the Adjusted R-squared values, as well as the low scores overall, tells us that none of the variables are actually great predictors. For that reason, let's just look at any one of the graphs, say Model 1. For the scatter plot, we can observe a distinct pattern that separates the values by clear intervals of 0.1. This makes sense given that the neighborhood ratings from 1-5 aren't continuous, rather scores rounded up to the 1st decimal place. The values range from around 2.4 to 3.5 and they don't closely follow the fitted line. All this indicates either one of two things: that the multiple linear regression is not the best algorithm for this dataset, or that the fundamental assumption that our predictor variables can actually explain the response variable is weak. This is further supported by the residual plot, which shows some type of pattern and clustering within the predicted values of 2.8 to 3.2. A good residual plot would have a more evenly scatter plot. The plots here with the most reassuring results are the Q-Q plots. Q-Q plots tell us about the validity of the assumption we make about the datasets distribution being normal such that a roughly straight indicates normal distribution. Earlier while cleaning the data, in order to fill in the empty values, I had to assume no skew and a pretty normal distribution. After sampling the dataset to create the "Core Citywide Survey" and "Key Commercial Areas" subsets, this is how I likely guaranteed the data's normality.

In terms of model performance, Model 5 technically had the highest Adjusted R-Squared. It had a score of 0.132, followed closely by Model 2 with a score of 0.131. Model 5 narrowed the dataset down to 10 variables while Model 2 narrowed it down to 7. Occam's Razor guides us to select the simpler model with fewer variables. With an almost negligible difference, this makes Model 2 a better model. The 7 final variables of Model 2 are: street_litter_dstr, gov_property_graffiti_count, broken_glass_street_sidewalk_dstr, feces_count, syringes_count, pavement_defects, and key_comm_area. If we go back and remind ourselves of the objective of this analysis, I am trying to understand which variables best indicate a neighborhood's rating. We'll also do well to be reminded that the most significant limitation in this project is that the neighborhood ratings were calculated in 2023 from a dataset completely different to the 2022 sidewalk and street report. Instead, the ratings were calculated through surveys that have an entirely different set of questions to gauge people's perceptions of the cleanliness and conditions of the street and sidewalk. To theorize what these variables mean, it would make most sense to understand how these 7 uncleanliness factors might affect any random person's perception of the street and sidewalk. The presence of street litter, broken glass, feces, and syringes is fairly straightforward. Graffiti on government property as opposed to other properties probably indicates neglect of an area by the government and increased rebellious sentiment, as opposed to those in

private property which is more expected. Similarly with pavement defects. The variable named Key Comm Area indicates whether an area is residential or commercial/industrial/mixed use. It makes sense that this would affect a person's perception of cleanliness as we can imagine this affects the layout of the area, frequency of use, visibility to the public, and overall attention from the government.

Conclusion

As evidenced by Figure 2, I found that I was able to mostly replicate all of the 2022 evaluation's main results, except for 1, with mostly small discrepancies most likely due to sampling differences. The one finding I couldn't replicate was about transit shelters due to a lack of proper instructions in the report nor in the metadata on how they calculate it. As for the regression analysis, none of the models were particularly strong, nor was any one model significantly better than the other. Model 2, made with a multiple linear regression filtering for variables with a $p<0.05$ after an initial multiple linear regression, was best, Occam's Razor considered. It had 7 variables that could explain the neighborhood rating if reasoned through the perspective of how an individual might perceive the cleanliness and condition of the streets and sidewalks. These variables describe street litter, the instances of graffiti on government property, broken glass, syringes, pavement defects, and whether the area was residential or not. Further studies should be made to expand this analysis. In response to the main limitations of this paper, one could try out different neighborhood ratings, comparing data that describe matching years, and trying out different regression algorithms to determine which variables are the best predictors.

References

SF Controller's Office. (2023a). *City survey: Streets and sidewalks*. San Francisco. https://sf.gov/data/city-survey-streets-and-sidewalks

SF Controller's Office. (2023b, April 13). *2023 City Survey Detailed Methodology*. https://sf.gov/sites/default/files/2023-04/2023%20City%20Survey%20Detailed%20Methodology.pdf

SF Controller's Office. (2023c, April 13). *City survey: Neighborhoods*. San Francisco. https://sf.gov/data/city-survey-neighborhoods#streets-and-sidewalks-ratings-by-neighborhood

SF Controller's Office. (2023d, May 22). *2022 Street & Sidewalk Maintenance Standards Report*. San Francisco. https://sf.gov/reports/may-2023/2022-street-sidewalk-maintenance-standards-report

SF Controller's Office. (2023e, May 22). *DPW street & sidewalk evaluation results, CY22*. City and County of San Francisco. https://data.sfgov.org/City-Infrastructure/DPW-Street-Sidewalk-Evaluation-Results-CY22/fsqv-4vqv

Smith, G. (2018). Step away from stepwise. *Journal of Big Data*, *5*(1), 1–12. https://doi.org/10.1186/s40537-018-0143-6

Appendix

```
# Ensure all files are in the same current working directory

import os

os.getcwd()

# Install then import packages using shorter aliases for efficiency

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

import geopandas as gpd

from shapely.geometry import Point, Polygon


pd.set_option('display.max_columns', None) # ensure dataframes display all columns

# 2022 Sidewalk Evaluation Dataset

# Source:

https://data.sfgov.org/City-Infrastructure/DPW-Street-Sidewalk-Evaluation-Results-CY22/fsqv-4vqv


# Consolidate geolocation variables into one 'geometry' column

eval_gdf = gpd.read_file('DPW_Street___Sidewalk_Evaluation_Results__CY22.csv')

eval_gdf['geometry'] = [Point(xy) for xy in zip(eval_gdf['x'], eval_gdf['y'])]

eval_gdf = eval_gdf.drop(['x', 'y', 'the_geom'], axis=1)

eval_gdf['geometry'] = eval_gdf['geometry'].apply(Point)


# Convert selected columns to integers or floats for analysis later

exclude_columns = ['Route Location:', 'Other dumped items:', 'Other sidewalk obstructions:', 'CreationDate',
```

```
'geometry']
int_columns = [col for col in eval_gdf.columns if col not in exclude_columns]
eval_gdf[int_columns] = eval_gdf[int_columns].apply(lambda col: pd.to_numeric(col, errors='coerce'),
axis=0)
# Import the dataset used for the evaluation
eval_gdf_raw = gpd.read_file('DPW_Street___Sidewalk_Evaluation_Results__CY22.csv')
# Check number of rows and columns
eval_gdf_raw.shape
# Preview a sample of the data
eval_gdf_raw.sample(n=3)
# Gleam a statistical understanding of the variables
eval_gdf_raw.describe(include='all')
# Create Core Citywide Dataset
core_citywide = eval_gdf.copy()
# Split the DataFrame into 80% residential
key_comm_areas_col = 'Is this route predominantly residential or commercial/industrial/mixed use?'

commercial_rows = eval_gdf[eval_gdf[key_comm_areas_col] == 1]
residential_rows = eval_gdf[eval_gdf[key_comm_areas_col] == 2]

# Calculate the number of samples needed for each category
total_samples = 1000
residential_samples = int(total_samples * 0.8)
commercial_samples = total_samples - residential_samples

# Sample from each category
```

```
residential_sample = residential_rows.sample(n=residential_samples, random_state=123)
commercial_sample = commercial_rows.sample(n=commercial_samples, random_state=123)

# Concatenate the samples
final_sample = pd.concat([residential_sample, commercial_sample])

# Shuffle the final sample to ensure randomness
core_citywide = final_sample.sample(frac=1, random_state=123)
# Create Key Commercial Areas subset
key_comm_areas_samples = 769
key_comm_areas = commercial_rows.sample(n=key_comm_areas_samples, random_state=123)
# Main function to replicate evaluation results
def replicate_survey_results(col, var_name,
                             core_df = core_citywide, # Default dataset unless otherwise specified
                             key_df = key_comm_areas, # Default dataset unless otherwise specified
                             subset = None,
                             drop_empty = True, draw_hist = True,
                             all_except_zero = False, all_except_one = False): # Core Citywide Survey and Key
Commercial Area Survey
    """
    Custom function that requires user to input a few things.
    col: Name of the column in strings (or passed as a variable)
    var_name: User must specify a name in snake case, ex: street_litter
    drop_empty: Boolean to drop empty values or not as that affects the final percentage
    draw_hist: Boolean to draw histogram of answers or not
    at_least_one: Boolean to determine if the criteria expects "at least one" instead of moderate to severe
```

```
    Outcome: Calculates percentage of matching criteria to overall dataset
    """
    # Create local variables to reference
    core_citywide = core_df.copy()
    key_comm_areas = key_df.copy()

    # Subset dataset dependent on another column having a value of 1
    if subset != None:
        core_citywide = core_citywide[core_citywide[subset]==1]
        key_comm_areas = key_comm_areas[key_comm_areas[subset]==1]

    # Drop empty values – can be turned off
    # If specified false, it means there were no empty values and no difference in the final values
    if drop_empty: # True by default
        core_citywide_dropped_count = len(core_citywide[core_citywide[col].isna()])
        core_citywide_dropped_pcnt = core_citywide_dropped_count/len(core_citywide) * 100
        core_citywide = core_citywide[-core_citywide[col].isna()]
        if core_citywide_dropped_count != 0:
            print(f"{core_citywide_dropped_count} rows or {core_citywide_dropped_pcnt:.2f}% of
core_citywide was empty")

        key_comm_areas_dropped_count = len(key_comm_areas[key_comm_areas[col].isna()])
        key_comm_areas_dropped_pcnt = key_comm_areas_dropped_count/len(key_comm_areas) * 100
        key_comm_areas = key_comm_areas[-key_comm_areas[col].isna()]
        if key_comm_areas_dropped_count != 0:
```

```
        print(f"{key_comm_areas_dropped_count} rows or {key_comm_areas_dropped_pcnt:.2f}% of
key_comm_areas was empty")


#     # Convert values from string to int for easier sorting
#     core_citywide[col] = core_citywide[col].astype(int)
#     key_comm_areas[col] = key_comm_areas[col].astype(int)


    # Modify criteria according to the specified column's score ranking system
    if all_except_zero:
        criteria = [num for num in core_citywide[col].unique() if num != 0]
    elif all_except_one:
        criteria = [num for num in core_citywide[col].unique() if num != 1]
    elif core_citywide[col].unique().max()>5:
        criteria = [num for num in core_citywide[col].unique() if num != 0]
    elif core_citywide[col].unique().max()==5: # scores of 3-5 represent moderate to severe
        criteria = [3, 4, 5]
    elif core_citywide[col].unique().max()==5: # scores of 3-5 represent moderate to severe
        criteria = [3, 4, 5]
    elif core_citywide[col].unique().max()==3: # scores of 2-3 represent moderate to severe
        criteria = [2, 3]
    elif core_citywide[col].unique().max()==1: # score of 1 represents presence vs absence
        criteria = [1]


    # Calculate percentages
    overall_percentage = len(core_citywide[core_citywide[col].isin(criteria)]) / len(core_citywide) * 100
    kca_percentage = len(key_comm_areas[key_comm_areas[col].isin(criteria)]) / len(key_comm_areas) * 100
```

```
# Print percentages
print(f"""
{overall_percentage:.2f}% {' '.join(var_name.split('_'))} in Core Citywide Survey
{kca_percentage:.2f}% {' '.join(var_name.split('_'))} in Key Commercial Areas
""")

# Draw histograms – can be turned off
if draw_hist: # True by default
    # Create a figure and a side-by-side grid of subplots
    fig, axes = plt.subplots(1, 2, figsize=(10, 5), sharey=True)

    # Plot histogram for core_citywide on the left
    core_citywide_counts = core_citywide[col].value_counts()[sorted(core_citywide[col].unique())]
    core_citywide_counts.plot(kind='bar', ax=axes[0], color='#0035a0')
    axes[0].set_title('Core Citywide')
    axes[0].set_ylabel('Frequency')
    axes[0].set_xlabel('')
    axes[0].tick_params(axis='x', rotation=0)

    # Plot histogram for key_comm_areas on the right
    key_comm_areas_counts = key_comm_areas[col].value_counts()[sorted(key_comm_areas[col].unique())]
    key_comm_areas_counts.plot(kind='bar', ax=axes[1], color='#3e7dfd')
    axes[1].set_title('Key Commercial Areas')
    axes[1].set_xlabel('')
    axes[1].tick_params(axis='x', rotation=0)
```

```
        # Adjust layout and display plot
        plt.suptitle(col)
        plt.tight_layout()
        plt.show()

    return round(overall_percentage), round(kca_percentage)
# Manually created csv
results_overview = pd.read_csv("2022_streets_sidewalks_cleanliness_survey_results_overview.csv")
results_overview.columns = ['Feature', 'Core Citywide Survey', 'Key Commercial Areas']

# Remove values after percentage symbol in 'Core Citywide Survey' column
results_overview['Core Citywide Survey'] = results_overview['Core Citywide Survey'].str.replace(r'%.*', '',
regex=True)

# Remove values after percentage symbol in 'Key Commercial Areas' column
results_overview['Key Commercial Areas'] = results_overview['Key Commercial Areas'].str.replace(r'%.*', '',
regex=True)

# Make sure both column values are floats
results_overview['Core Citywide Survey'] = results_overview['Core Citywide Survey'].astype(float)
results_overview['Key Commercial Areas'] = results_overview['Key Commercial Areas'].astype(float)

# Add empty columns for replicated values to compare later
results_overview['Core Citywide Survey (Replicated)'] = ''
results_overview['Key Commercial Areas (Replicated)'] = ''
```

```
# Display overview of main results in report
results_overview

# Create variable to represent the long column name written in the form of a question
street_litter = 'Select the statement that best describes the amount and distribution of litter in the street.'

# Perform replication, save results onto table, and display results
results_overview.loc[0,[core, key]] = replicate_survey_results(street_litter, 'street_litter')
# Create variable to represent the long column name written in the form of a question
sidewalk_litter = 'Select the statement that best describes the amount and distribution of litter on the
sidewalk.'

# Perform replication, save results onto table, and display results
results_overview.loc[1,[core, key]] = replicate_survey_results(sidewalk_litter, 'sidewalk_litter')
# Create variable to represent the long column name written in the form of a question
trash_overflow = 'Are any trash receptacles overflowing?'

# Perform replication, save results onto table, and display results
results_overview.loc[2,[core, key]] = replicate_survey_results(trash_overflow, 'trash_overflow')
# Create variable to represent the long column name written in the form of a question
sidewalk_obstruction = 'Is sidewalk clearance less than 4 feet wide or 8 feet high at any point?'

# Perform replication, save results onto table, and display results
results_overview.loc[3,[core, key]] = replicate_survey_results(sidewalk_obstruction, 'sidewalk_obstruction')
# Create variable to represent the long column name written in the form of a question
```

```
pavement_defects = 'How severe are the pavement defects?'

# Perform replication, save results onto table, and display results
results_overview.loc[4,[core, key]] = replicate_survey_results(pavement_defects, 'pavement_defects')
# Create variable to represent the long column name written in the form of a question
illegal_dumping = 'How many large abandoned items are present?'

# Perform replication, save results onto table, and display results
results_overview.loc[5,[core, key]] = replicate_survey_results(illegal_dumping, 'illegal_dumping')
# Create variables to represent the long column names written in the form of questions
graffiti_instances = 'How many instances of graffiti are present on private property or on the sidewalk or
non-painted curb immediately adjacent to private property?'
graffiti_instances1 = 'How many instances of graffiti are present on SF gov property?'
graffiti_instances2 = 'How many instances of graffiti are present on other property?'

graffiti_groups = [graffiti_instances, graffiti_instances1, graffiti_instances2]
# Create a new max_graffiti column and dataset to fit the format for the replication function
core_citywide_max_graffiti = core_citywide.copy()
key_comm_areas_max_graffiti = key_comm_areas.copy()
core_citywide_max_graffiti['max_graffiti_instance'] = core_citywide[graffiti_groups].max(axis=1) # takes the
max of 3 values
key_comm_areas_max_graffiti['max_graffiti_instance'] =
key_comm_areas_max_graffiti[graffiti_groups].max(axis=1) # takes the max of 3 values

# Define the bucket ranges and labels
bins = [0, 25, 100, float('inf')]
```

```
labels = [1, 2, 3]

# Create a new column 'bucketed_max_graffiti' with bucket labels
core_citywide_max_graffiti['max_graffiti_instance'] =
pd.cut(core_citywide_max_graffiti['max_graffiti_instance'].astype(int), bins=bins, labels=labels)
key_comm_areas_max_graffiti['max_graffiti_instance'] =
pd.cut(key_comm_areas_max_graffiti['max_graffiti_instance'].astype(int), bins=bins, labels=labels)

# Drop NaN values from the 'max_graffiti_instance' column
core_citywide_max_graffiti = core_citywide_max_graffiti.dropna(subset=['max_graffiti_instance'])
key_comm_areas_max_graffiti = key_comm_areas_max_graffiti.dropna(subset=['max_graffiti_instance'])
# Perform replication, save results onto table, and display results
results_overview.loc[12,[core, key]] = replicate_survey_results('max_graffiti_instance', 'max_graffiti_instance',
                    core_df=core_citywide_max_graffiti,
                    key_df=key_comm_areas_max_graffiti)
# Create variable to represent the long column name written in the form of a question
broken_glass = 'Select the statement that best describes the amount and distribution of broken glass on the
street and sidewalk:'

# Perform replication, save results onto table, and display results
results_overview.loc[6,[core, key]] = replicate_survey_results(broken_glass, 'broken_glass',
all_except_one=True)
# Create variable to represent the long column name written in the form of a question
feces = 'How many instances of feces are present?'

# Perform replication, save results onto table, and display results
```

```
results_overview.loc[7,[core, key]] = replicate_survey_results(feces, 'feces', all_except_zero=True)

# Filter Results Overview table to only include the values computed for

exclude_from_results_overview = [8, 9, 10, 11, 13]

final_results_overview = results_overview.drop(exclude_from_results_overview)


# Set the index to 'Feature' column

final_results_overview.set_index('Feature', inplace=True)

final_results_overview.astype(float)

# Set the width of the bars

bar_width = 0.35


# Create the figure and axes for the plots

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5), sharey=True)


# Plot 1: Core Citywide Survey vs Core Citywide Survey (Replicated)

plot1_df = final_results_overview[['Core Citywide Survey', 'Core Citywide Survey (Replicated)']]

plot1_df.plot(kind='bar', ax=axes[0], width=bar_width, color=['blue', 'red'])

axes[0].set_title('Core Citywide Survey vs Core Citywide Survey (Replicated)', size=10)

axes[0].set_xlabel('Feature')

axes[0].set_ylabel('Percentage')

axes[0].set_xticklabels(plot1_df.index, size=6, rotation=90)

axes[0].set_ylim(0, 100)

axes[0].legend()


# Plot 2: Key Commercial Areas vs Key Commercial Areas (Replicated)

plot2_df = final_results_overview[['Key Commercial Areas', 'Key Commercial Areas (Replicated)']]
```

```
plot2_df.plot(kind='bar', ax=axes[1], width=bar_width, color=['blue', 'red'])
axes[1].set_title('Key Commercial Areas vs Key Commercial Areas (Replicated)', size=10)
axes[1].set_xlabel('Feature')
axes[1].set_xticklabels(plot2_df.index, size=6, rotation=90)
axes[1].set_ylim(0, 100)
axes[1].legend()

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
# SF Analysis Neighborhoods
# Source:
https://data.sfgov.org/Geographic-Locations-and-Boundaries/Analysis-Neighborhoods/p5b7-5n3h
analysis_gdf = gpd.read_file('analysis_neighborhoods.shp')
analysis_gdf.head(n=3)
# Plot the polygons and points
fig, ax = plt.subplots(figsize=(8, 6))
analysis_gdf.plot(ax=ax, color='blue', edgecolor='black', alpha=0.7, label='Polygons')
eval_gdf.sample(n=450).plot(ax=ax, color='red', markersize=20, label='Points')
plt.xticks(rotation=45)  # Rotate x-axis labels by 45 degrees
ax.set_title('2022 Sampled Streets by SF Neighborhoods')
plt.show()

# Exclude columns that don't have clear instructions on how to interpret their values
```

```
# Exclude columns that are not indicators of an area's cleanliness
exclude_columns = ['ObjectID', 'Route Location:', 'Route ID:', 'Other dumped items:', 'If yes, identify the cause (select all that apply)', 'Other sidewalk obstructions:', 'Which of the following issues are present in or on any of the transit shelters along the route?','CreationDate', 'SF Find Neighborhoods 2', 'x', 'y', 'the_geom', 'Current Supervisor Districts 2', 'geometry']
selected_columns = [col for col in core_citywide.columns if col not in exclude_columns]
reg_gdf = core_citywide[selected_columns].copy()

reg_gdf.head(n=1)
# Simplify columns into snake case variable-like names
reg_gdf_cols = [
    'street_litter_dstr',
    'street_litter_obst',
    'sidewalk_litter_dstr',
    'large_abandoned_items_count',
    'items_type',
    'has_illegal_dump_obstruct_street',
    'private_property_graffiti_count',
    'gov_property_graffiti_count',
    'other_property_graffiti_count',
    'broken_glass_street_sidewalk_dstr',
    'broken_glass_street_sidewalk_count',
    'feces_count',
    'syringes_count',
    'condoms_count',
    'dead_animals_count',
```

```
    'has_trash_receptacles',
    'has_trash_receptacle_overflow',
    'sidewalk_clearance_obstructed',
    'has_sidewalk_unmarked_defects',
    'has_sidewalk_marked_defects',
    'pavement_defects',
    'has_foul_odors',
    'odor_from_sewer_or_basin',
    'has_transit_shelter',
    'key_comm_area',
    'Analysis Neighborhoods 2'
]
# Change column names for easier viewing and manipulation
reg_gdf.columns = reg_gdf_cols
# Analysis Neighborhoods 2 (Manually confirmed by graphing points and referring to location in SF gov
map)
# Source:
https://data.sfgov.org/Geographic-Locations-and-Boundaries/Analysis-Neighborhoods/p5b7-5n3h
neighborhood_dict = {
    1: "Bayview",
    2: "Bernal Heights",
    3: "Haight Ashbury",
    4: "Mission Bay",
    5: "Castro",
    6: "Chinatown",
    7: "Excelsior",
```

8: "Financial District/South Beach",

9: "Hayes Valley",

10: "Glen Park",

11: "Inner Richmond",

#12: "Golden Gate Park",

13: "Marina",

14: "Inner Sunset",

#15: "Japantown",

16: "Lakeshore",

#17: "Lincoln Park",

18: "Lone Mountain/USF",

20: "Mission",

21: "Nob Hill",

22: "Noe Valley",

23: "North Beach",

24: "Oceanview/Merced/Ingleside",

25: "Portola",

26: "Potrero Hill",

28: "Outer Mission",

29: "Outer Richmond",

30: "Pacific Heights",

31: "Presidio Heights",

32: "Russian Hill",

33: "Seacliff",

34: "South of Market",

35: "Sunset/Parkside",

```
    36: "Tenderloin",
    38: "Twin Peaks",
    39: "Western Addition",
    40: "Visitacion Valley",
    41: "West of Twin Peaks"
}
# Analysis Neighborhoods 2 Scores
# Source: https://sf.gov/data/city-survey-streets-and-sidewalks

# "We filtered neighborhoods with fewer than 10 responses from results
# "because results aren't reliable with that few responses, to maintain accuracy:
neighborhood_scores = {
    "Bayview": 2.9,  # "Bayview Hunters Point"
    "Bernal Heights": 3.0,
    "Haight Ashbury": 3.2,
    "Mission Bay": 3.0,
    "Castro": 3.0,  # "Castro/Upper Market"
    "Chinatown": 3.5,
    "Excelsior": 3.2,
    "Financial District/South Beach": 2.8,
    "Hayes Valley": 3.0,
    "Glen Park": 3.4,
    "Inner Richmond": 2.8,
    #"Golden Gate Park": 3.0, # excluded
    "Marina": 3.0,
    "Inner Sunset": 3.3,
```

```
    #"Japantown": 3.0, # excluded
    "Lakeshore": 2.9,
    #"Lincoln Park": 2.9, # excluded
    "Lone Mountain/USF": 3.1,
    "Mission": 2.9,
    "Nob Hill": 3.0,
    "Noe Valley": 3.3,
    "North Beach": 3.1,
    "Oceanview/Merced/Ingleside": 3.4,
    "Portola": 3.1,
    "Potrero Hill": 3.2,
    "Outer Mission": 3.0,
    "Outer Richmond": 2.8,
    "Pacific Heights": 3.0,
    "Presidio Heights": 3.4,
    "Russian Hill": 3.0,
    "Seacliff": 2.7,
    "South of Market": 2.7,
    "Sunset/Parkside": 3.1,
    "Tenderloin": 2.3,
    "Twin Peaks": 3.1,
    "Western Addition": 2.8,
    "Visitacion Valley": 2.9,
    "West of Twin Peaks": 3.2
}
# Drop rows where 'Analysis Neighborhoods 2' is not in neighborhood_dict
```

```
reg_gdf_rating = reg_gdf[reg_gdf['Analysis Neighborhoods 2'].isin(neighborhood_dict)].copy()

# Map the values in 'Analysis Neighborhoods 2' to neighborhood names using neighborhood_dict
reg_gdf_rating['Analysis Neighborhoods 2'] = reg_gdf_rating['Analysis Neighborhoods
2'].map(neighborhood_dict)

# Create the new column 'neighborhood_street_and_sidewalk_rating'
reg_gdf_rating['neighborhood_street_and_sidewalk_rating'] = reg_gdf_rating['Analysis Neighborhoods
2'].map(neighborhood_scores)

# Drop 'Analysis Neighborhoods 2' as we won't be needing it for the regression
reg_gdf_rating = reg_gdf_rating.drop(columns=['Analysis Neighborhoods 2'])
# Create a new dataset that has the NaN values filled in each column with the corresponding mean value
df = reg_gdf_rating.fillna(reg_gdf_rating.mean()).copy()
# Define the response column
y_variable = 'neighborhood_street_and_sidewalk_rating'

# Calculate correlation matrix using pandas method
corr_matrix = df.drop(y_variable, axis=1).corr()

# Create correlation plot
plt.figure(figsize=(30, 20))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', square=True, annot_kws={'fontsize': 10})
plt.title('Correlation Matrix')
plt.show()
import pandas as pd
```

```
import statsmodels.api as sm

# Define the response column
y_variable = 'neighborhood_street_and_sidewalk_rating'

# Define the response variable and predictor variables
X1 = df.drop(columns=[y_variable])
y = df[y_variable]

# Add a constant term to the predictor variables
X1 = sm.add_constant(X1)

# Fit the multivariable regression model
model1 = sm.OLS(y, X1).fit()

# Get the results of the regression results
results1 = model1.summary()
results1
# Extract p-values from Model 1
p_values1 = model1.pvalues

# Set a significance level (e.g., 0.05)
significance_level = 0.05

# Select predictor variables based on Model 1's p-values
X2_variables = p_values1[p_values1 < significance_level].index
```

```
X2 = X1[X2_variables]

# Fit Model 2 with selected predictor variables
model2 = sm.OLS(y, X2).fit()

# Get the results of the Model 2
results2 = model2.summary()
results2
# Define the response variables
X3 = df.drop(columns=[y_variable, 'sidewalk_litter_dstr', 'other_property_graffiti_count',
'gov_property_graffiti_count'])

# Add a constant term to the predictor variables
X3 = sm.add_constant(X3)

# Fit the multivariable regression model
model3 = sm.OLS(y, X3).fit()

# Get the results of Model 3
results3 = model3.summary()
results3
# Extract p-values from Model 3
p_values3 = model3.pvalues

# Select predictor variables based on Model 3's p-values
X4_variables = p_values3[p_values3 < significance_level].index
```

```
X4 = X3[X4_variables]

# Fit the Model 4 with selected predictor variables
model4 = sm.OLS(y, X4).fit()

# Get the summary of the Model 4
results4 = model4.summary()
results4
# Define your response variable (y) and predictor variables (X)
X5 = df.drop(columns=[y_variable])

# Add a constant term to the predictor variables
X = sm.add_constant(X5)

# Initialize the list of selected predictor variables
selected_vars = list(X.columns)

# Perform backward stepwise regression
max_adj_r2 = -float('inf')  # Initialize with negative infinity
final_model = None
loop_count = 0  # Initialize the loop counter

while len(selected_vars) > 0:
    # Fit the model
    model5 = sm.OLS(y, X[selected_vars]).fit()
```

```
    # Calculate adjusted R-squared
    adj_r2 = model5.rsquared_adj

    # Check if the current model has a higher adjusted R-squared
    if adj_r2 > max_adj_r2:
        max_adj_r2 = adj_r2
        worst_var = None

        # Find the predictor with the highest p-value
        worst_var = model5.pvalues.idxmax()

        # Remove the worst predictor from the selected list
        selected_vars.remove(worst_var)
    else:
        break

    loop_count += 1  # Increment the loop counter

# Print the final model summary
print(model5.summary())

# Print the number of loops
print(f"Number of loops: {loop_count}")
# Create a 3x5 subplot layout
fig, axes = plt.subplots(3, 5, figsize=(15, 9))
fig.tight_layout(pad=3.0)
```

```
# Iterate through models
for i, model in enumerate([model1, model2, model3, model4, model5]):
    # Scatter plot with line of best fit
    ax = axes[0, i]
    y_pred = model.predict()
    ax.scatter(y, y_pred, alpha=0.5)
    ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)
    ax.set_title(f'Model {i+1} Scatter Plot')
    ax.set_xlabel('Actual')
    ax.set_ylabel('Predicted')

    # QQ Plot
    ax = axes[1, i]
    sm.qqplot(model.resid, line='s', ax=ax)
    ax.set_title(f'Model {i+1} QQ Plot')

    # Residual Plot
    ax = axes[2, i]
    ax.scatter(y_pred, model.resid, alpha=0.5)
    ax.axhline(y=0, color='r', linestyle='--', lw=2)
    ax.set_title(f'Model {i+1} Residual Plot')
    ax.set_xlabel('Predicted')
    ax.set_ylabel('Residual')

# Display the plots
```

```
plt.show()

# List of models

models = [model1, model2, model3, model4, model5]


# Calculate R-squared for each model and store in the DataFrame

rsquared_adj_values = []

for model in models:

    rsquared_adj = model.rsquared_adj

    rsquared_adj_values.append(rsquared_adj)


rsquared_adj_df = pd.DataFrame({'Adj. R-squared': rsquared_adj_values})


# Find the best model based on R-squared

best_model_value = sorted(rsquared_adj_df['Adj. R-squared'].copy())[-1]

best_model = rsquared_adj_df[rsquared_adj_df['Adj. R-squared'] == best_model_value].index[0]


# Find the second best model based on R-squared

second_best_model_value = sorted(rsquared_adj_df['Adj. R-squared'].copy())[-2]

second_best_model = rsquared_adj_df[rsquared_adj_df['Adj. R-squared'] ==

second_best_model_value].index[0]


# Display the Adj. R-squared values for all models

print("Adj. R-squared values for each model:")

rsquared_adj_df.index = [f"Model {i+1}" for i in range(len(models))]

print(rsquared_adj_df)
```

```
# Print the best model
print("Best model based on Adj. R-squared: Model", best_model + 1)
print("Second best model based on Adj. R-squared: Model", second_best_model + 1)
# Plot the Adj. R-squared values for visual comparison
plt.figure(figsize=(10, 6))
ax = sns.barplot(x=rsquared_adj_df.index, y=rsquared_adj_df['Adj. R-squared'], color='skyblue')
plt.ylim(0, 1)  # Set y-axis limits to max 1
plt.xlabel('Model')
plt.ylabel('Adj. R-squared')
plt.title('Adj. R-squared Values by Model')

# Add labels to the bars
for i, v in enumerate(list(rsquared_adj_df['Adj. R-squared'])):
    ax.text(i, v + 0.02, f'{v:.3f}', ha='center', color='red')
plt.show()
```