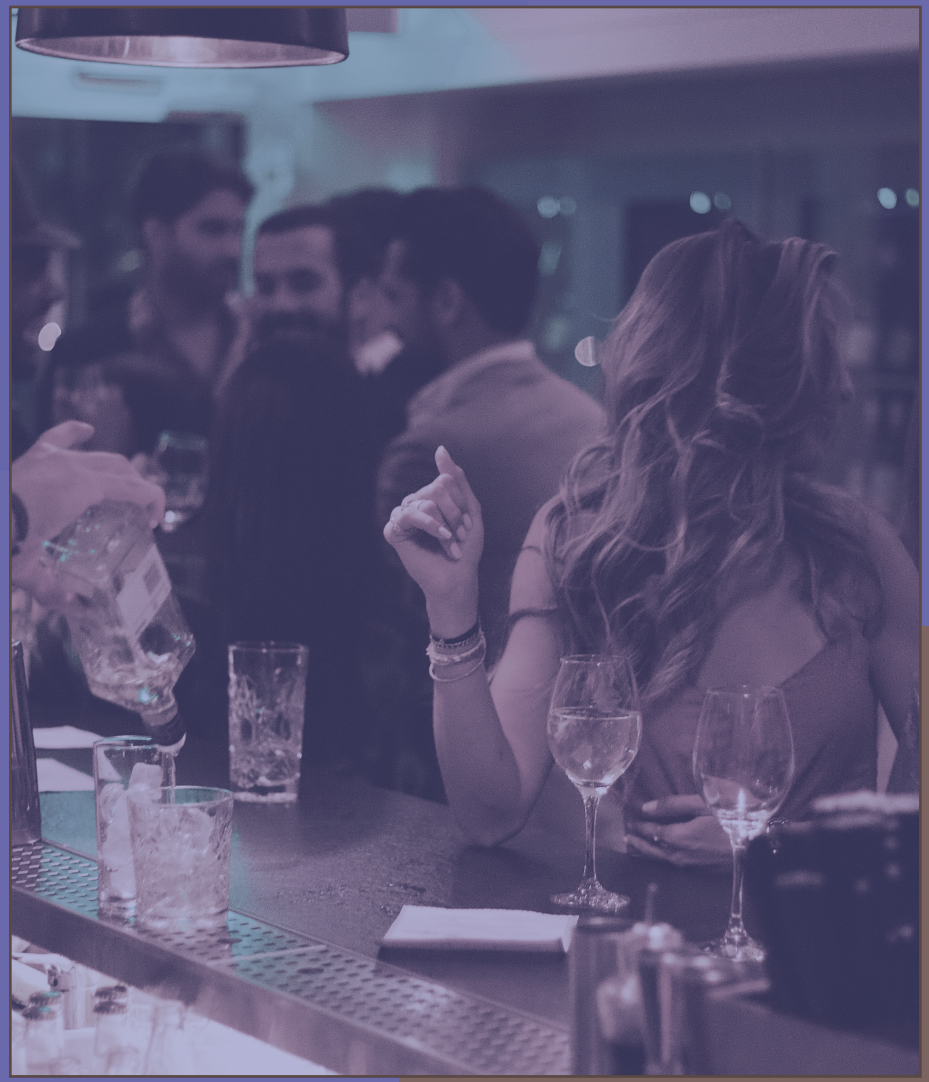
DISCOVERING GOOD AND SAFER PUBS FOR WOMEN WITH DATA SCIENCE

Danielle Pinho
May, 2022

OUTLINE



ABSTRACT

London is well-known for its pubs. With more than 3500 pubs in the city, it can be expected to be part of a Londoner life. It is not different for Londoner women, as they represent 48% of all pubs consumers. But women might be uncomfortable and feel unsafe to be out at night in pubs, since pubs can be male-dominant environments. To address this issue, this project aims to predict good and safer pubs for women.

In order to achieve this goal, data about London's pubs and their surroundings were collected, manipulated, analyzed and prepared for prediction though clustering models. Finally, the prediction was made and the clusters evaluated.

It was possible to find the best pubs, with good rating, easy access by subway, close to police station and average crime rate.

INTRODUCTION

In the 2019 Cultural Infrastructure Report, the Greater London Authority informed that London had 3540 pubs, with most of them (62.7%) located in inner London boroughs (Greater London Authority, 2019).

Boroughs with most pubs:

- City of Westminster: 430
- Camden: 230
- Islington: 215

Boroughs with lowest pubs:

- Barking and Dagenham: 20
- Merton, Newham and Waltham: 45 each

It is uncertain how the covid-19 pandemics affected these numbers, since a new report is still to be released.

INTRODUCTION

Between the years of 2010 and 2014, only 24% of women in the UK visited pubs weekly, while for the same period, up to 44% of men did (Statista, 2014).
In a research conducted by 888Poker in 2018, which included 2000 UK-based women, it was shown that a third of those women felt that pubs are male-orientated in a negative way and over half of them affirmed that sexist behaviour was the most intimidating conduct in a male-dominated environment (Thatcher, 2020).

Nonetheless, women presence have increased since then. As for 2019, 48,7% of all pub consumers were women (Leader, 2020).
Also, more women are drinking beer in a weekly basis. The SIBA British Craft Beer Report 2020 pointed a increase of 11% in beer consumption by women in just a year, from 2019 to 2020 (The University Caterers Organisation, 2020).

VIOLENCE AGAINST WOMEN

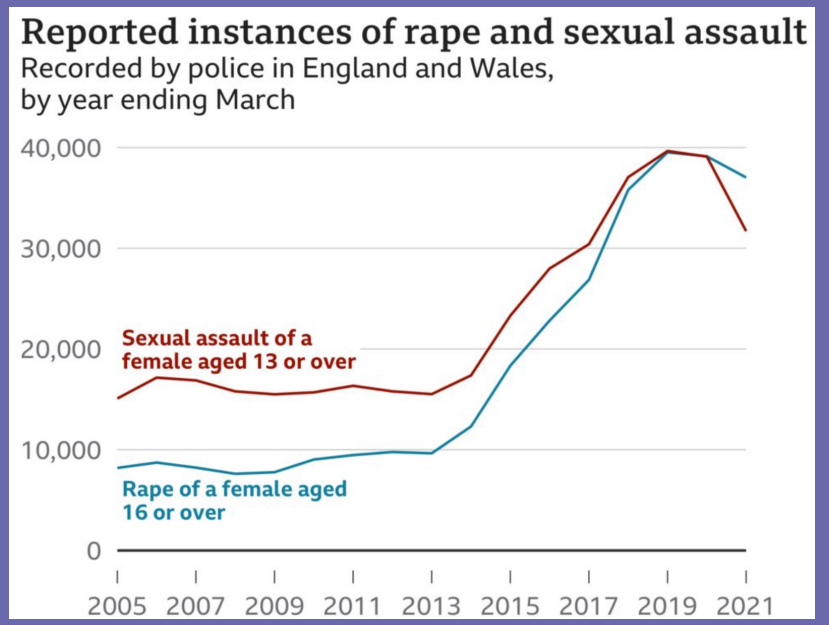
In September 2021, it was reported that 40.572 women were victim of sexual assault, an increase of 13% from 2020 and the highest number ever recorded, and 41.332 rape crimes, the highest figure recorded to date, as well (Gilder; Clarke, 2022).

It is believed that this number is even greater, since many women do not report sexual assaults to the police, "for reasons ranging from societal cultures of victim-blaming, to myths and stereotypes that impact how survivors are treated." (Andrea Simon apud Gilder; Clarke, 2022).

For 2021, the numbers of reported crimes against women were (BBC, 2022):

- Rape: 63.136
- Sexual offences: 170.973

Source: Greater London Authority, 2022



Source: Gilder; Clarke, 2022

QUESTIONS AND GOALS

IS IT POSSIBLE TO FIND GOOD PUBS IN LONDON WHERE WOMEN CAN FEEL SAFE TO DRINK AND ENJOY THEMSELVES?

THIS PROJECT AIMS TO PREDICT THE BEST AND SAFER PUBS FOR WOMEN TO VISIT, USING CLUSTERING MODELS.

METHODOLOGY

DATA COLLECTION

To enable this project, data about London's pubs and its surroundings were needed.

Pubs and police stations were collected based in the latitude and longitude of each borough, within a radius of 3000m, at Foursquare API. Crime data was filtered by a specific type of crime, as explained in the next slides.

Place of collection:	Data needed:	Data collected:
• Web scrapping – London boroughs • Foursquare API • Web scrapping – London underground • UK Police CSV files	• London boroughs • Pubs • Police stations • Subway stations • Crime data	• London boroughs: 33 • Pubs: 1728 • Police stations: 470 • Subway stations: 304 • Crime data: 102.297

WEB SCRAPPING – LONDON BOROUGHS

To obtain data about London's pubs, first, a list with all borough names and their latitude, longitude coordinates were needed.

The official boroughs were found at the London Council website and a request to get the information was made.

Through BeautifulSoup the response was parsed and the table containing the borough list was transformed into a Pandas dataframe.

Source: London Council, 2022

```
response = requests.get("https://directory.londoncouncils.gov.uk/")

soup = BeautifulSoup(response.text, "html.parser")
london_table = soup.find("table", {"class": "table"})

london_df = pd.read_html(str(london_table))
london_df = pd.DataFrame(london_df[0])
```

FOURSQUARE API

The Foursquare API was used 3 times: requesting pubs core data, police station data and pubs detailed data.

The core pubs data and police stations data were collected from the Places endpoint. To request data from that endpoint, several functions were created.

First, a request was made for each borough through a loop and since each request has a limit of 50 venues, another loop was made to access the next pages with more venues.

```python
# make request to fetch venues data from Foursquare API
def make_request(url):
    token = config("TOKEN")
    headers = {"Accept": "application/json", "Authorization": f"{token}"}
    response = requests.request("GET", url, headers=headers)
    return response


# get next url from Foursquare API response
def get_next_url(response):
    try:
        return re.findall("<(.*?)>", response.headers["Link"])[0]
    except (KeyError, IndexError):
        return None


# get venues data from a given coordinate for the given categories
def get_nearby_venues_data(coordinate, categories):
    json_list = []
    url = f"https://api.foursquare.com/v3/places/search?ll={coordinate}&radius=3000&ca

    while url != None:
        response = make_request(url)
        json_file = response.json()
        json_list.append(json_file)

        url = get_next_url(response)
        time.sleep(0.1)
        print(url)
        print(response.headers)

    return json_list


# get venues data of given categories for all the coordinates of London's boroughs
def get_venues_data(categories):
    venues_data = []
    for coordinate in coordinates.values():
        nearby_venues = get_nearby_venues_data(coordinate, categories)
        venues_data.extend(nearby_venues)
    return venues_data
```

FOURSQUARE API – DETAILED DATA

After getting the pubs data and parsing it into a pandas dataframe, a new request to the Foursquare API was made for each pub in the dataframe to fetch information about price, popularity and rating.

A new function was created for this purpose.

This information was also parsed into a new dataframe.

```
# get rating, popularity and price range data of given venues from Foursquare API
def get_detailed_data(dataframe):
    json_list = []
    token = config("TOKEN")
    for id in dataframe.fsq_id:
        url = f"https://api.foursquare.com/v3/places/{id}?fields=fsq_id%2Crating%2

        headers = {"Accept": "application/json", "Authorization": f"{token}"}

        response = requests.request("GET", url, headers=headers)
        json_file = response.json()
        json_list.append(json_file)
        print(response.text)
    return json_list


# collection pubs detailed data
json_list = get_detailed_data(pubs_df)
```

WEB SCRAPPING – LONDON UNDERGROUND

The Foursquare API did not returned the correct number of subway stations in London. Therefore, another source was needed.

To collect information about the subway stations, a web scrapping was performed at the OpenStreetMap Wiki page. As the previous web scrapping, BeautifulSoup was used to parse the response data and a dataframe was made containing all London's Tube stations.

```
# get data from Wiki Open Street Map page about London Underground
response = requests.get(
    "https://wiki.openstreetmap.org/wiki/List_of_London_Underground_stations"
)
soup = BeautifulSoup(response.text, "html.parser")

# select the first table in the webpage
underground_table = soup.find("table", {"class": "wikitable"})

# transform table into a pandas dataframe
underground_df = pd.read_html(str(underground_table))
underground_df = pd.DataFrame(underground_df[0])
```

	name	latitude	longitude	line
0	Acton Town	51.502500	-0.278126	District, Piccadilly
1	Acton Central	51.508835	-0.263033	London Overground
2	Acton Central	51.508560	-0.262880	London Overground
3	Aldgate	51.513940	-0.075370	Metropolitan
4	Aldgate East	51.515140	-0.071780	District, Hammersmith & City
5	Alperton	51.540970	-0.300610	Piccadilly
6	Amersham	51.674350	-0.607320	Metropolitan
7	Angel	51.532530	-0.105790	Northern
8	Archway	51.565360	-0.134740	Northern
9	Arnos Grove	51.616250	-0.133550	Piccadilly

CRIME DATA – CSV FILES

Crime data from London were collected from the UK Police Data website. It consisted in csv files where each contained a month of crime information in a specific jurisdiction. For this project, data about the last 6 available months (February 2022 to September 2021) concerning the Metropolitan Police and City of London Police was gathered.

After merging all files, the dataframe was filtered to contain only violence and sexual offences and rows with missing latitude, longitude coordinates were dropped.

```
import glob
import pandas as pd

# Through glob, transform all data crime csv files into a single pandas dataframe
files = glob.glob("/content/*.csv")
frames = [pd.read_csv(file) for file in files]
df = pd.concat(frames)

# Check if dataframe is correct
print(df.shape)
print(df.head())

# Filter dataframe for a specific type of crime and assign this to a new dataframe
crimes_df = df[df["Crime type"] == "Violence and sexual offences"]
print(crimes_df.shape)

# Drop rows without latitude and longitude data
crimes_df.dropna(subset=["Longitude", "Latitude"], inplace=True)
```

DATA WRANGLING

When collecting data at Foursquare API, specific fields regarding the venues were selected. The json file returned had the data of these fields in dictionary format, as seen in the picture below.

	fsq_id	categories	geocodes	location	name
0	4bc1e42eabf49521c690c193	[{'id': 13018, 'name': 'Pub', 'icon': {'prefix...	{'main': {'latitude': 51.653075, 'longitude': ...	{'address': '92 Wood St', 'admin_region': 'Eng...	The Black Horse
1	4b995bccf964a5209f7535e3	[{'id': 13018, 'name': 'Pub', 'icon': {'prefix...	{'main': {'latitude': 51.652979, 'longitude': ...	{'address': '58 High St', 'admin_region': 'Eng...	Ye Olde Mitre Inne
2	4bb4da3fa7059521b8cc1bce	[{'id': 13018, 'name': 'Pub', 'icon': {'prefix...	{'main': {'latitude': 51.652533, 'longitude': ...	{'address': 'Barnet Rd', 'admin_region': 'Engl...	The Arkley
3	4dc436e5ae608779d11bd561	[{'id': 13018, 'name': 'Pub', 'icon': {'prefix...	{'main': {'latitude': 51.650059, 'longitude': ...	{'address': '3 East Barnet Rd', 'admin_region'...	Railway Tavern
4	4d72b1e78e12b1f793863f05	[{'id': 13018, 'name': 'Pub', 'icon': {'prefix...	{'main': {'latitude': 51.653499, 'longitude': ...	{'address': '84 High St', 'admin_region': 'Eng...	The Kings Head

It was necessary to extract all relevant data from this dictionary and create new proper columns.

Before extracting the data, duplicated venues were checked and dropped.

Of the 1728 pubs found, 569 were duplicated and of the 470 police stations, 224 were duplicated.

DATA WRANGLING

To extract the relevant data, a function was created to avoid code repetition and generalize its use.

After extracting, it was noticed that some venues have missing values, including the geocodes variable.

To find the latitude and longitude of those venues, Nominatim from geopy library was used and the coordinates were found through the venues' addresses or postcodes.

Finally, the values in geocodes column were separated into two new columns: latitude, longitude.

Geocodes, Categories and Location columns that would not be used anymore were dropped.

```
# extract selected data from column with values in dictionaries
def extract_value(df, column, regex):
    return df[column].str.extract(regex)
```

```
pubs_df["geocodes"] = extract_value(pubs_df, "geocodes", "{'main':.\{(.*?)\}")
pubs_df["address"] = extract_value(pubs_df, "location", "{'address':.'(.*?)',")
pubs_df["locality"] = extract_value(pubs_df, "location", "'locality':.'(.*?)',")
pubs_df["neighborhood"] = extract_value(pubs_df, "location", "'neighborhood':.\['(.*)'\]")
pubs_df["postcode"] = extract_value(pubs_df, "location", "'postcode':.'(.*?)'")
pubs_df["category"] = extract_value(pubs_df, "categories", "'name':.'(.*?)'")
```

```
# try to find coordinates from address
def get_geocodes_by_street(row):
    address = row["address"]
    try:
        location = geolocator.geocode(f"{address}, London")
        row[
            "geocodes"
        ] = f"'latitude': {location.latitude}, 'longitude': {location.longitude}"
    except:
        get_geocodes_by_postcode(row)


# try to find coordinates from postcode
def get_geocodes_by_postcode(row):
    id = row["fsq_id"]
    try:
        location = geolocator.geocode(row["postcode"])
        row[
            "geocodes"
        ] = f"'latitude': {location.latitude}, 'longitude': {location.longitude}"
    except:
        row["geocodes"] = np.nan
        print(f"It was not possible to find coordinates for {id}")
```

DATA WRANGLING – MERGING DETAILED DATA

The detailed pubs data were stored in a different csv file, therefore it was necessary to merge its dataframe into the core pubs dataframe.

First, it was verified that both dataframes had the same size in axis 0.

Then, the merge was successfully made with left join using the fsq_id column as reference.

The image below shows the first 5 records of the dataframe after the merge.

	fsq_id	name	address	locality	neighborhood	postcode	category	latitude	longitude	popularity	price	rating
0	4bc1e42eabf49521c690c193	The Black Horse	92 Wood St	London	Barnet	EN5 4BW	Pub	51.653075	-0.206657	0.939551	1.0	8.1
1	4b995bccf964a5209f7535e3	Ye Olde Mitre Inne	58 High St	Hertfordshire	Barnet	EN5 5SJ	Pub	51.652979	-0.199367	0.982768	1.0	7.6
2	4bb4da3fa7059521b8cc1bce	The Arkley	Barnet Rd	Barnet	Barnet	EN5 3EP	Pub	51.652533	-0.219573	0.987375	1.0	7.7
3	4dc436e5ae608779d11bd561	Railway Tavern	3 East Barnet Rd	Barnet	Barnet	EN4 8RR	Pub	51.650059	-0.174692	0.950895	1.0	7.4
4	4d72b1e78e12b1f793863f05	The Kings Head	84 High St	Barnet	Barnet	EN5 5SN	Pub	51.653499	-0.200979	0.377777	1.0	6.6

MISSING DATA

With all the collected data in one dataframe, missing values were checked. The number of missing values were particularly high for rating and neighborhood.

Neighborhood data would not be used since its values were not consistent with the official boundaries, therefore, it would be substituted for boroughs data later in the project.

Regarding rating, since it did not have any correlation with other variables, it was decided to drop the rows with missing values, as there was still plenty of pubs information.

There were 16 missing values for price and were filled with its mode.

And there were only 5 missing values for postcode, therefore they were filled manually.

After dealing with missing data, the pubs dataframe had 865 entries.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 865 entries, 0 to 1157
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   name        865 non-null    object 
 1   postcode    865 non-null    object 
 2   latitude    865 non-null    object 
 3   longitude   865 non-null    object 
 4   popularity  865 non-null    float64
 5   price       865 non-null    float64
 6   rating      865 non-null    float64
dtypes: float64(3), object(4)
memory usage: 86.4+ KB
```

DATA TIDYING – NUMBER OF CRIMES

To find the number of crimes that occurred near the pubs, a function was built. This function receives 3 parameters (latitude, longitude and a dataframe), iterates through the latitude and longitude of each crime and gets the distance between the crime and a pub. The distance was calculated in meters, using the haversine library.

If the distance is lesser than 500m, a crimes' counter created for each pub is incremented by one.

This function was called inside the apply function that creates a new column in the pubs dataframe to store the total of crimes returned.

```python
def get_number_of_crimes(latitude1, longitude1, df):
    loc1 = (latitude1, longitude1)
    crimes = 0
    for latitude2, longitude2 in list(zip(df["latitude"], df["longitude"])):
        loc2 = (latitude2, longitude2)
        distance = hs.haversine(loc1, loc2, unit=Unit.METERS)
        if distance <= 500:
            crimes += 1
    return crimes
```

```python
pubs_data["number_of_crimes"] = pubs_data.apply(
    lambda row: get_number_of_crimes(row["latitude"], row["longitude"], crime_data),
    axis=1,
)
```

DATA TIDYING – DISTANCE TO SUBWAY AND POLICE

To find if a pub has nearby subway and police stations, a function was built to return the smallest distance between a pub and a venue.

The function receives 3 parameters (latitude, longitude and a dataframe), iterates through a list of coordinates, calculate the distance in meters between two points using haversine and returns the smallest value.

New columns were created to store the distance between a pub and the venue (distance_metro; distance_police).

Also, two other columns were created in the pubs dataframe to inform if a pub has a subway/police within the distance of 500m. If it has, the number 1 would be assigned and if not, the number assigned would be 0.

```python
#calculate the distance between two venues and get the smallest distance
def get_distance(latitude1, longitude1, df):
    loc1 = (latitude1, longitude1)
    distance_min = 1000000
    for latitude2, longitude2 in list(zip(df.latitude, df.longitude)):
        loc2 = (latitude2, longitude2)
        distance = hs.haversine(loc1, loc2, unit=Unit.METERS)
        if distance < distance_min:
            distance_min = distance
    return distance_min
```

```python
pubs_data["metro"] = pubs_data["distance_metro"].apply(lambda x: 1 if x < 500 else 0)
```

```python
pubs_data["police"] = pubs_data["distance_police"].apply(lambda x: 1 if x < 500 else 0)
```

DATA TIDYING – NUMBER OF SUBWAY AND POLICE

To get the number of nearby subway and police stations for each pub, a new request was made to the Foursquare API. Since the Foursquare API did not have enough information about London's subway stations, Google Places API was used instead.

The functions requested to the API information about metro/police stations within 500m of each pub's coordinates. The length of the json response was assigned as the value for new columns in the pubs dataframe (number_of_metro; number_of_police).

Unfortunately, Google Places API also did not returned all the available subway stations in London.

```python
def request_police_foursquare(latitude, longitude):
    try:
        coordinate = f"{latitude},{longitude}"
        token = config("TOKEN")
        headers = {"Accept": "application/json", "Authorization": f"{token}"}
        url = f"https://api.foursquare.com/v3/places/search?ll={coordinate}&radius=500
        response = requests.request("GET", url, headers=headers)
        json_file = response.json()
        print(json_file)
        count = len(json_file["results"])
        return count
    except:
        print(coordinate)
        print(response)
        count = np.nan
        return count
```

```python
def request_metro_google(latitude, longitude):
    try:
        coordinate = f"{latitude},{longitude}"
        secret_key = config("SECRET_KEY")
        url = f"https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=
        payload = {}
        headers = {}

        response = requests.request("GET", url, headers=headers, data=payload)
        print(response.text)
        json_file = response.json()
        count = len(json_file["results"])
        return count
    except:
        print(coordinate)
        print(response)
        count = np.nan
        return count
```

DATA TIDYING – LONDON BOROUGHS

Since the information gathered at Foursquare API regarding location was inconsistent, to get the right borough of a pub a new web scrapping was necessary. For this task, Selenium was used, since the webpage requires clicking to access the intended information.

The webpage used was a government official page that displays the proper borough from a postcode.

The code below returns a list with the pubs' boroughs. If a postcode is incorrect, the value would be assigned as numpy NAN.

Four postcodes presented errors and were fixed manually.

After this, 6 pubs outside London were found and dropped.

It was possible to gather pubs from 31 of the 33 boroughs.

```
borough_list = []
for postcode in postcode_list:
  try:
    browser.get("https://www.gov.uk/find-local-council")
    browser.find_element_by_id("postcode").clear()
    browser.find_element_by_id("postcode").send_keys(postcode)
    browser.find_element_by_css_selector(".gem-c-button.gem-c-button--bottom-margin").click()
    time.sleep(0.5)
    borough = browser.find_element_by_css_selector(".local-authority-results p.govuk-body:first-child").text
    #print(borough)
    borough_list.append(borough)
  except:
    print(postcode)
    borough_list.append(np.nan)
```

DATA TIDYING – FINAL DATAFRAME

After Data Wrangling and Tidying stages, the final dataframe, to be used in analysis and modelling, has 15 columns and 859 rows.

	name	postcode	borough	latitude	longitude	price	rating	popularity	number_of_crimes	number_of_metro	number_of_police	distance_metro	distance_police	metro	police
0	The Black Horse	EN5 4BW	London Borough of Barnet	51.653075	-0.206657	1.0	8.1	0.939551	30	0	2.0	904.722178	390.281423	0	1
1	Ye Olde Mitre Inne	EN5 5SJ	London Borough of Barnet	51.652979	-0.199367	1.0	7.6	0.982768	45	1	3.0	443.945965	119.561106	1	1
2	The Arkley	EN5 3EP	London Borough of Barnet	51.652533	-0.219573	1.0	7.7	0.987375	2	0	0.0	1767.173867	1215.815963	0	0
3	Railway Tavern	EN4 8RR	London Borough of Barnet	51.650059	-0.174692	1.0	7.4	0.950895	64	0	0.0	1343.248049	1149.621255	0	0
4	The Kings Head	EN5 5SN	London Borough of Barnet	51.653499	-0.200979	1.0	6.6	0.377777	40	0	3.0	568.292759	98.112834	0	1

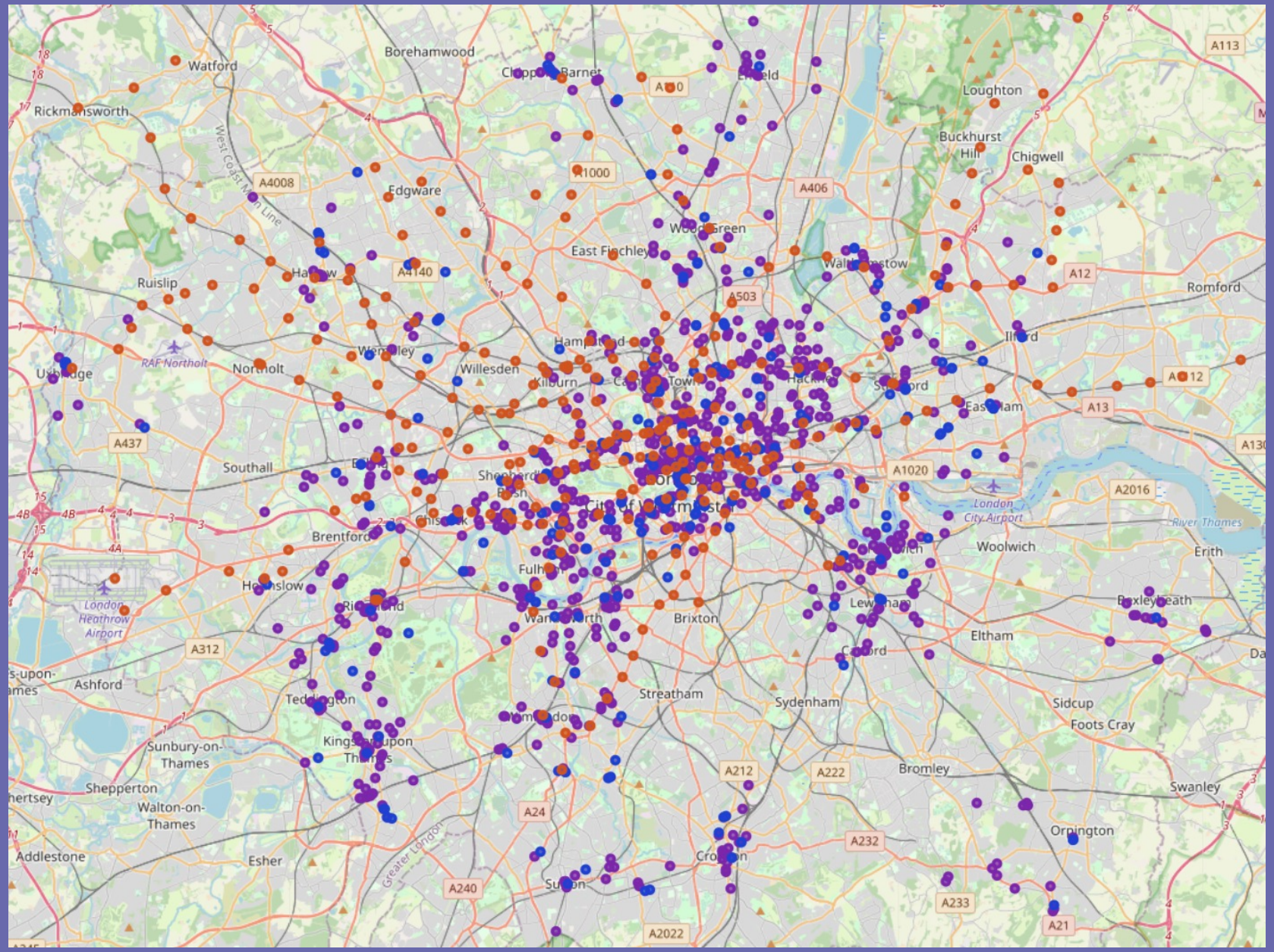
FINAL DATAFRAME - DATA DICTIONARY

Name	Description	Type
name	The best-known name for the FSQ Place.	object
postcode	Pub's postcode.	object
borough	Name of the pub's administrative division region.	object
latitude	Pub's latitude.	float
longitude	Pub's longitude.	float
price	A numerical value that best describes the pricing tier of the FSQ Place, based on known prices for menu items and other offerings. Values include: 1 = Cheap; 2 = Moderate; 3 = Expensive; 4 = Very Expensive.	float
rating	A numerical rating (from 0.0 to 10.0) of the FSQ Place, based on user votes, likes/dislikes, tips sentiment, and visit data.	float
popularity	A measure of the FSQ Place's popularity, by foot traffic. This score is on a 0 to 1 scale and uses a 6-month span of POI visits for a given geographic area.	float
number_of_crimes	Total number of violence and sexual crimes occurred in a 500m radius of a pub.	int
number_of_metro	Total number of subway stations found in a 500m radius of a pub, gathered at Google Places API.	int
number_of_police	Total number of police stations found in a 500m radius of a pub, gathered at Foursquare API.	float
distance_metro	The distance from a pub to the nearest subway station.	float
distance_police	The distance from a pub to the nearest police station.	float
metro	A numerical value that describes the presence of a subway station within a radius of 500m from a pub. Values are: 1- has metro; 0- does not have subway.	int
police	A numerical value that describes the presence of a police station within a radius of 500m from a pub. Values are: 1- has police station; 0- does not have police station.	int

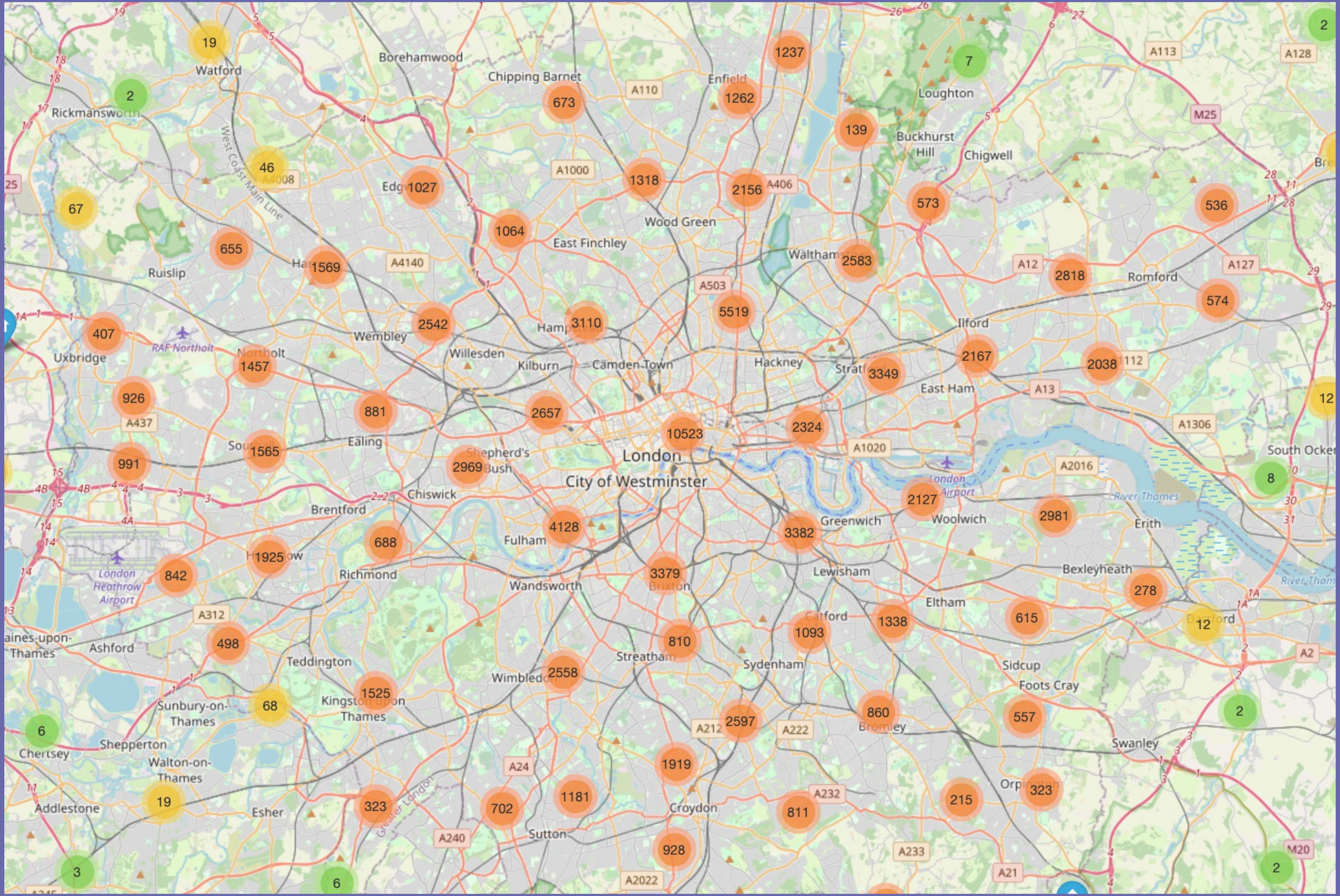
EXPLORATORY DATA ANALYSIS

LONDON'S PUBS, SUBWAYS AND POLICE STATIONS

- Pubs
- Tube stations
- Police stations

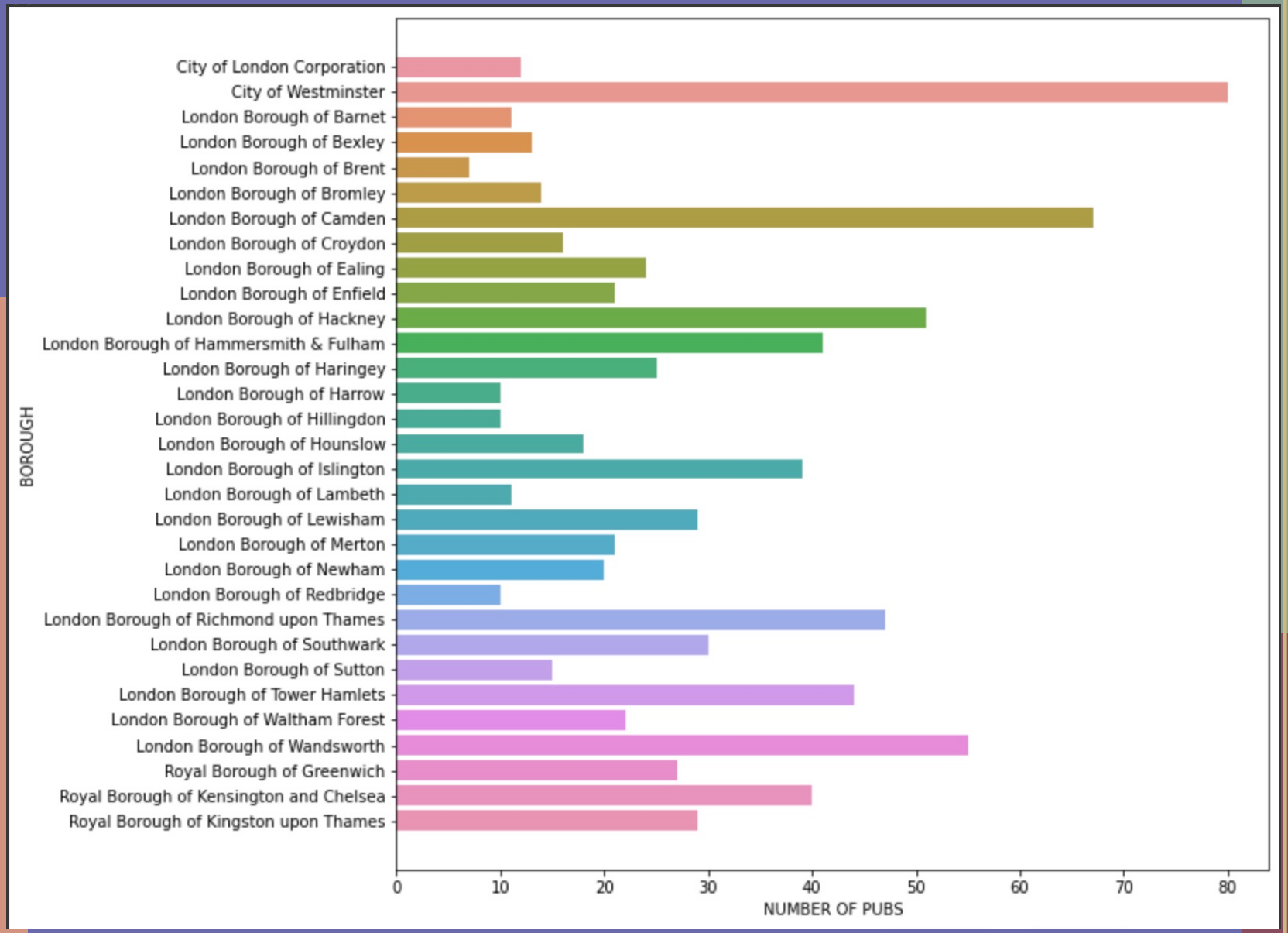


VIOLENCE AND SEXUAL OFFENCES IN LONDON AREA



PUBS PER BOROUGH

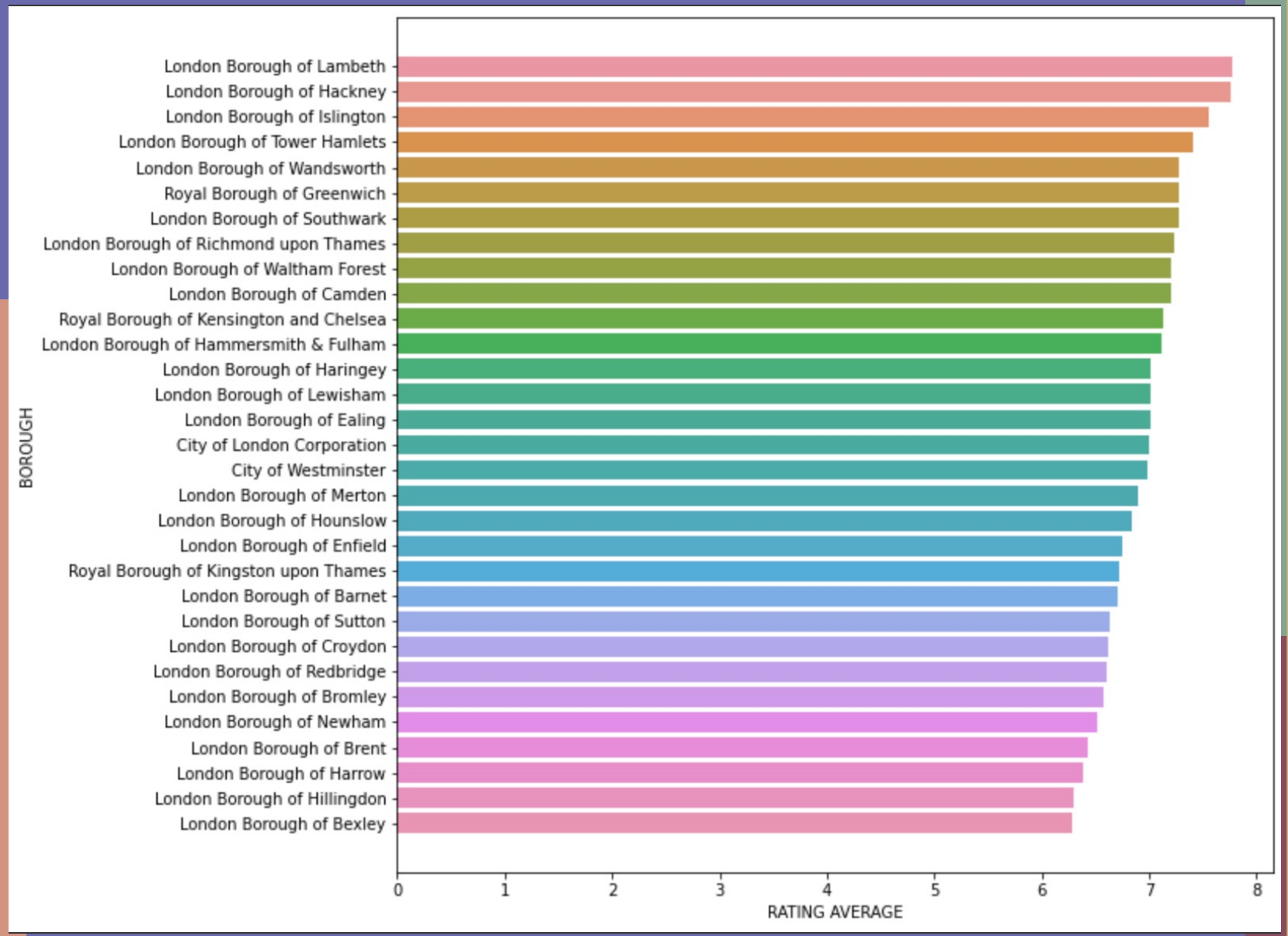
As reported in the introduction, the data collected for this project shows that the borough with most pubs is City of Westminster, followed by Camden.



RATING PER BOROUGH

The average rating distribution has a variance of 1.5 between boroughs, ranging from 6.27 to 7.77.

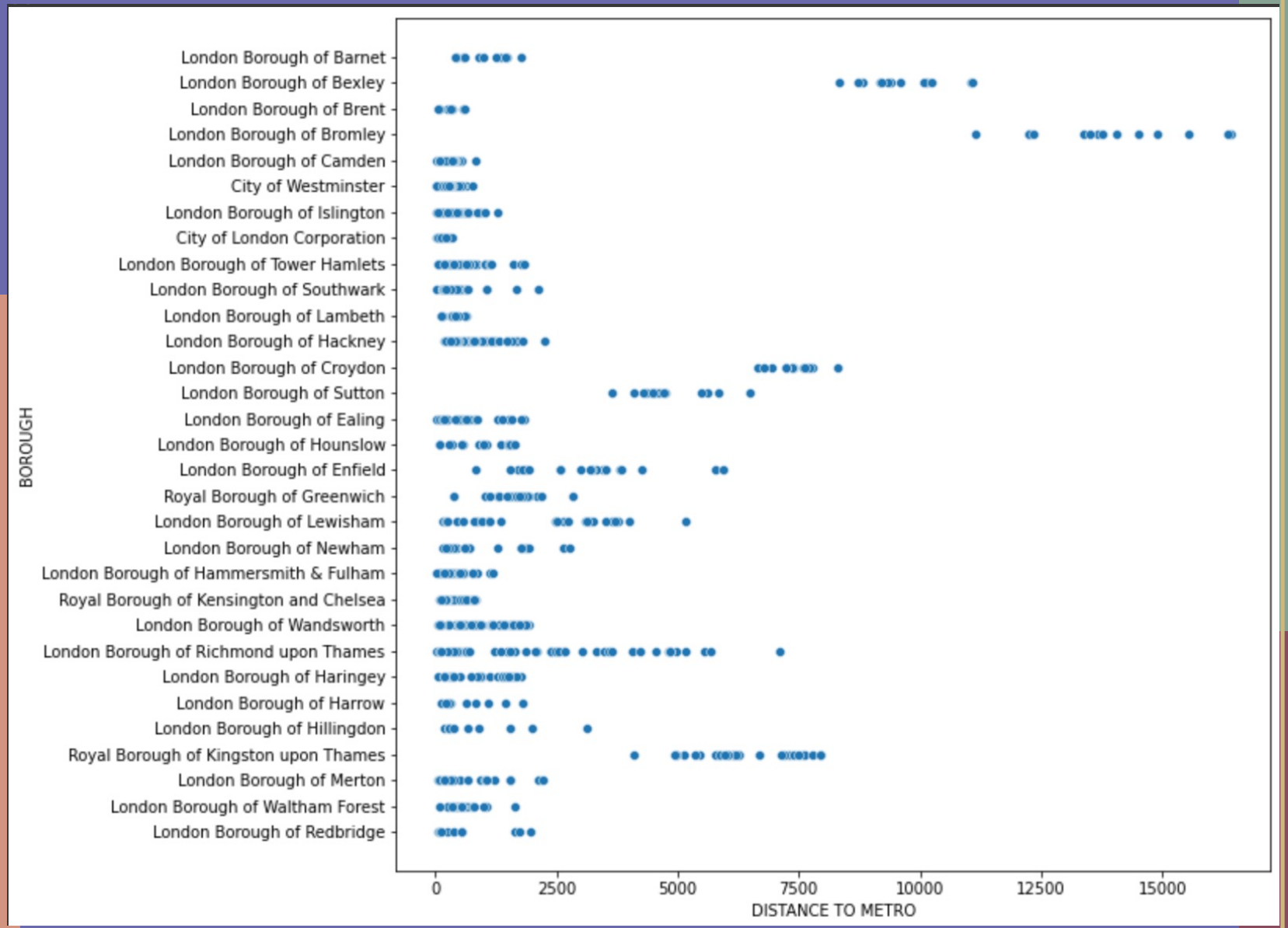
City of Westminster has an average of 6.98 and Camden has an average of 7.2.



DISTANCE TO METRO

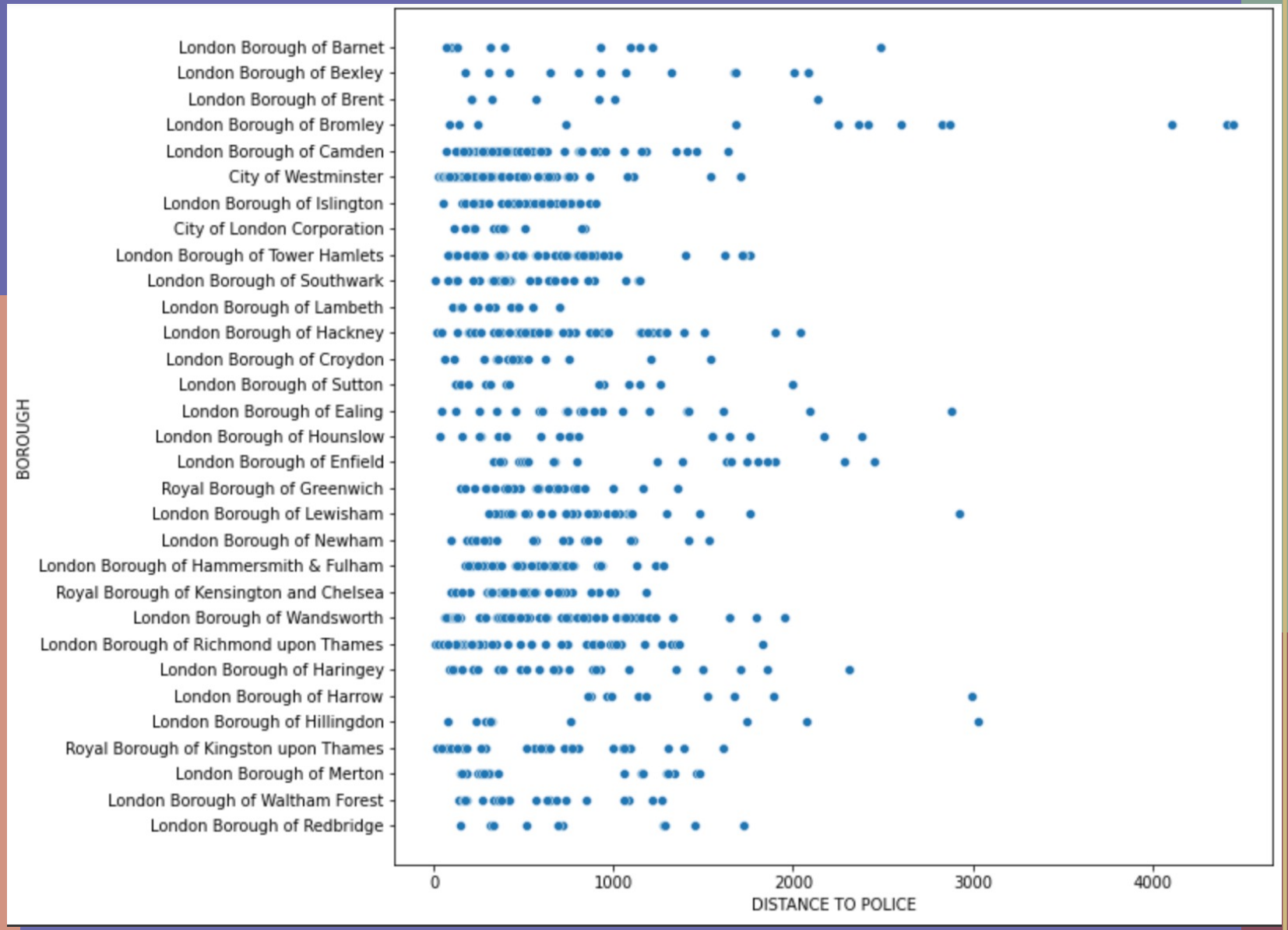
Most boroughs have pubs that are close to a subway station, but Bexley, Bromley, Croydon, Sutton and Kingston upon Thames do not.

This finding is consistent with the offer of subway stations for those boroughs.



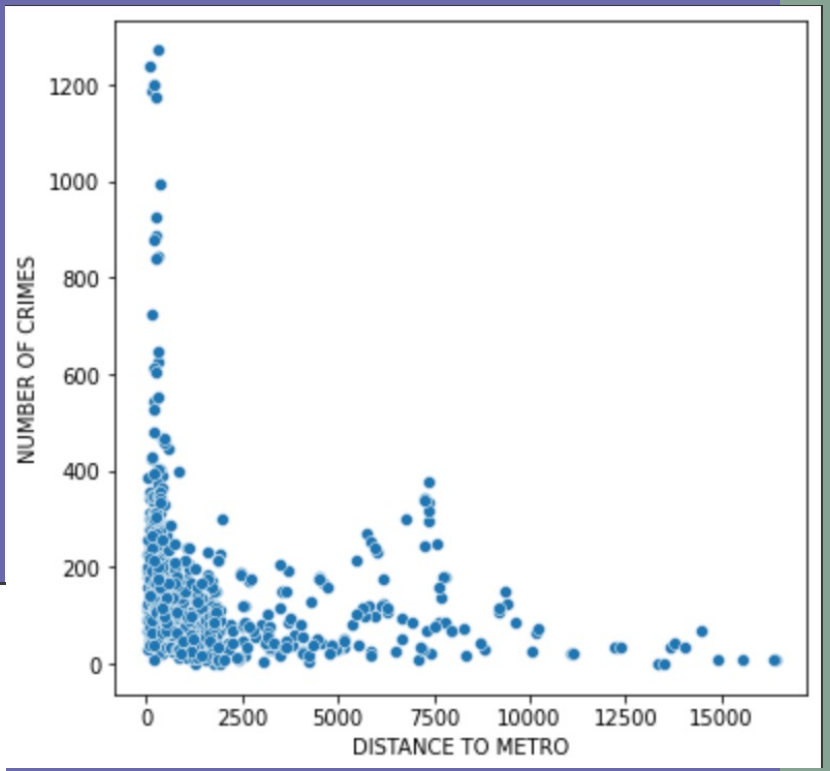
DISTANCE TO POLICE STATION

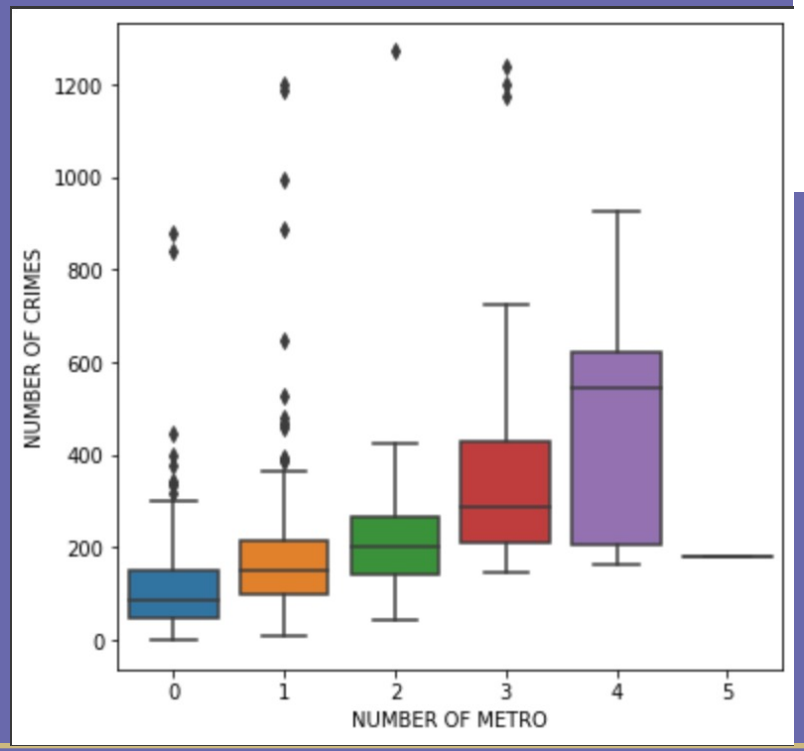
Most all the boroughs have pubs that have nearby police stations, with the exception of Harrow.



RELATION BETWEEN NUMBER OF CRIMES AND SUBWAY

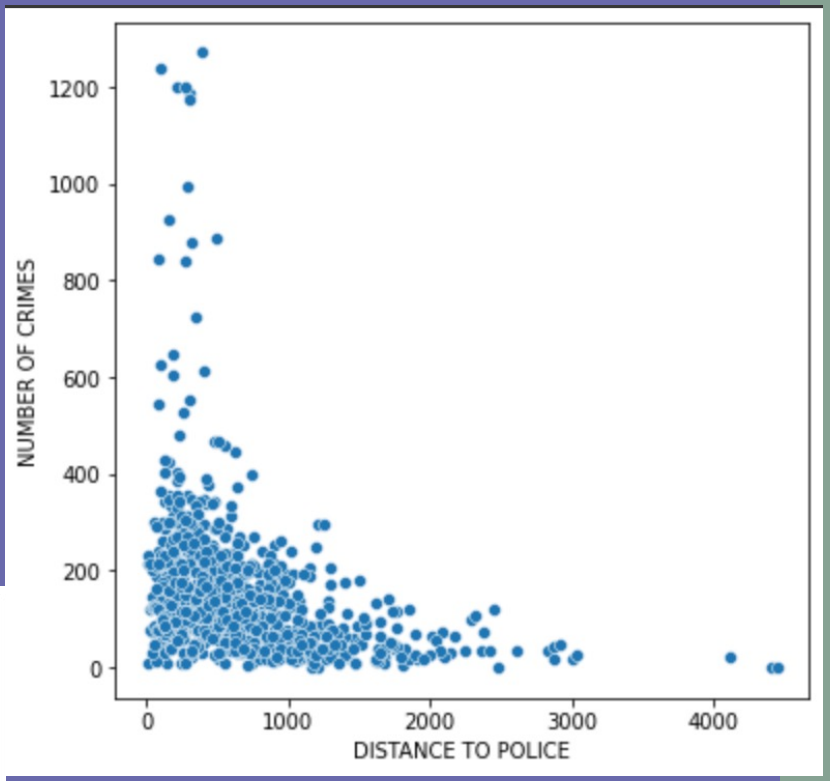
It is possible to notice that the pubs that have the greatest number of crimes are nearby subway stations.
Considering the number of subway stations, it is possible to observe that the pub that have more subway stations nearby also have more crimes nearby.

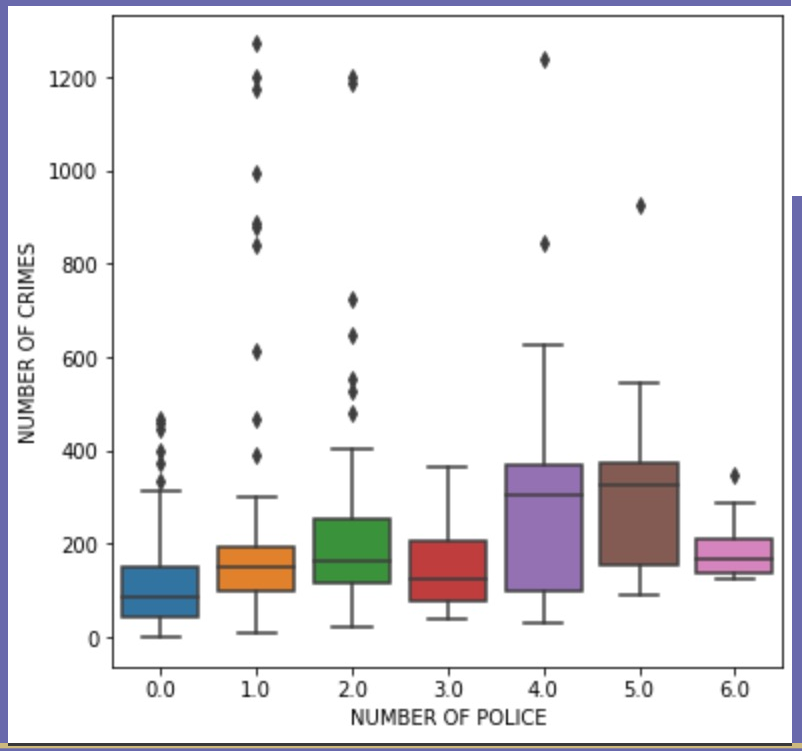




RELATION BETWEEN NUMBER OF CRIMES AND POLICE

The same pattern observed between subway stations and number of crimes cannot be seem between police stations and number of crimes.





PREDICTIVE ANALYSIS

DATA MODELLING

To be able to predict the best pubs for women, this project used clustering algorithms.

The algorithm chosen was K-Means, imported from the Scikit-learn library.

The final dataframe have 15 variables, but some variables are derived from the same subject, such as subway and police.

Because of that, several subsets of variables were explored to find the best features for modelling.

The feature chosen, at first, were: *price, rating, popularity, number_of_crimes, metro and police*.

During modelling, it was observed that the variable price was being assigned a greater importance than the other variables and the clusters were being reduced to their price range. Therefore, to avoid this, this variable was removed from the features.

```
pubs_data[["price", "rating", "popularity", "number_of_crimes", "metro", "police"]].corr()
```

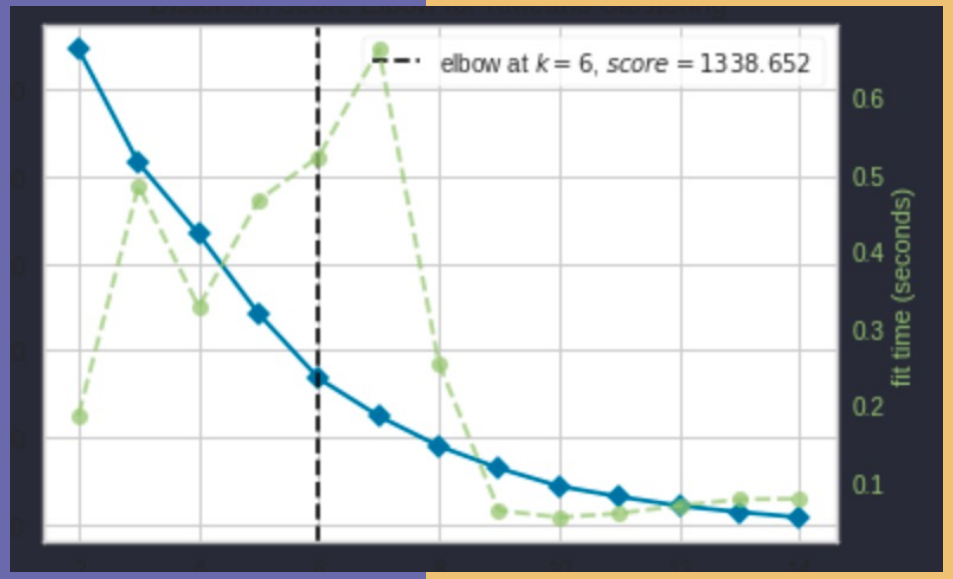
	price	rating	popularity	number_of_crimes	metro	police
price	1.000000	0.241572	0.124505	0.203373	0.182665	0.070721
rating	0.241572	1.000000	0.157893	-0.020165	-0.063078	-0.079769
popularity	0.124505	0.157893	1.000000	0.103772	0.138168	-0.009793
number_of_crimes	0.203373	-0.020165	0.103772	1.000000	0.374413	0.321621
metro	0.182665	-0.063078	0.138168	0.374413	1.000000	0.258249
police	0.070721	-0.079769	-0.009793	0.321621	0.258249	1.000000

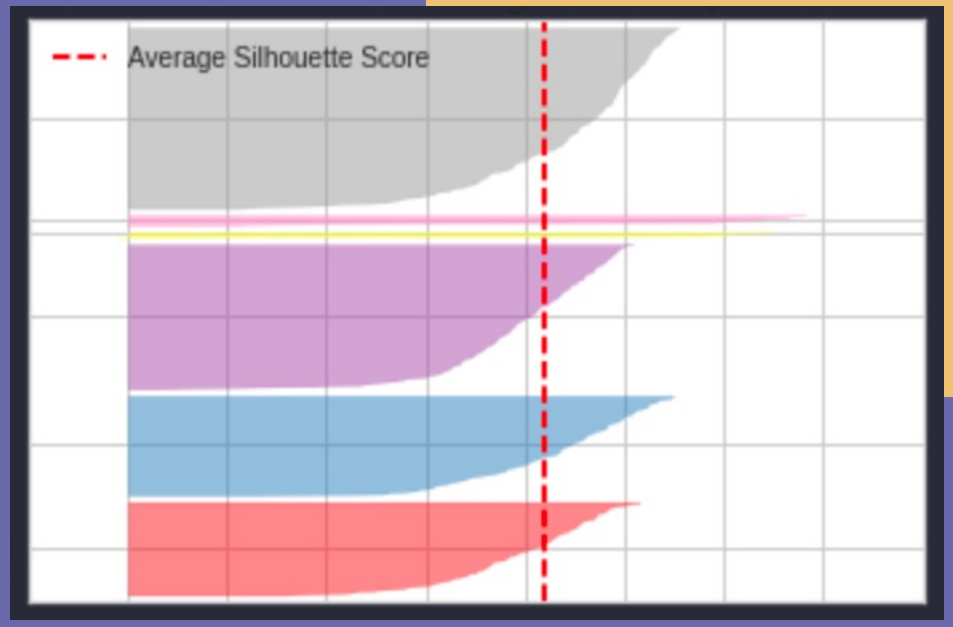
FINDING THE BEST K

To find the best K, a k elbow visualizer was used from the Yellowbrick library. The best k found, in a range from 2 to 15 (not included) was k = 6.

The silhouette was also analyzed for the same range. Although 6 k did provide good clusters for this project problem, it did not brought the best pubs regarding the metrics desired. Therefore, the number of clusters was increased until the best pubs were found.

As k was being increased, it was possible to notice that the cluster with the best pubs was about the same.



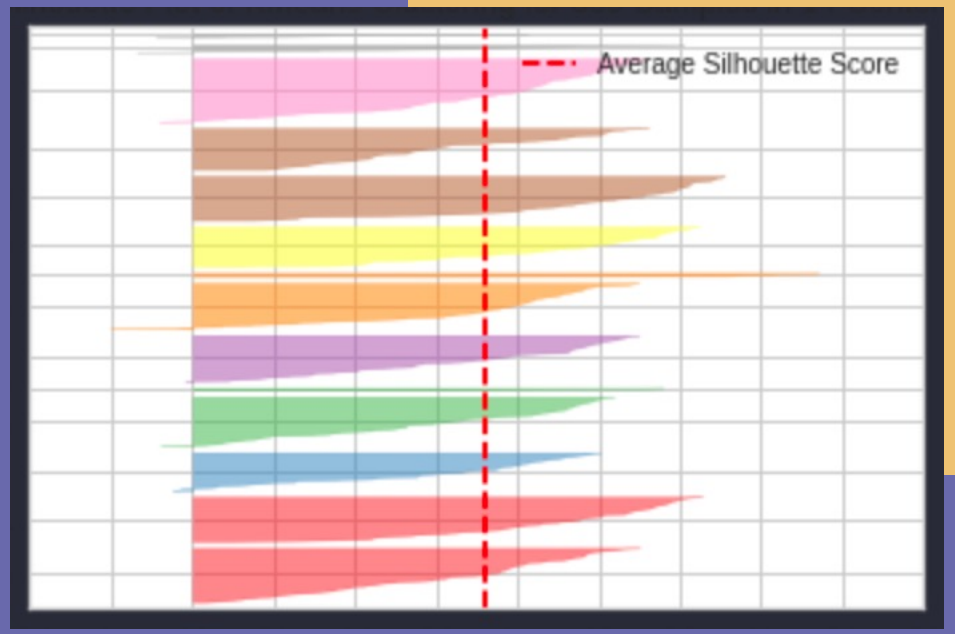


FINDING THE BEST PUBS

The metrics established for best pubs were:

- good rating: above general median (7.1)
- Popular: above general median (0.96)
- Number of crimes: lower general median (120)
- At least 1 subway and 1 police station within 500m

These metrics were found with 14 clusters. The silhouette for this k is shown in the image beside.
This group consists in 74 pubs and all of them were part at the same cluster when k was equal to 6.



THE BEST PUBS

As the image below shows, this cluster presents all the desired metric: high rating (the average is 7.6 and the lowest rating is still close to the general median), very popular, with low crimes nearby and within a 500m distance to a subway and police stations.

It was not possible to find places close to subway with all values in number of crimes below 120 (the general median), since number of crimes and the metro station have some correlation.

```python
# high rating, popular, low crime, close to metro and police (best group)
pubs_data[pubs_data["cluster_nk"] == 13].describe()
```

Python

	latitude	longitude	price	rating	popularity	number_of_crimes	number_of_metro	number_of_police	distance_metro	distance_police	metro	police	cluster_6k	cluster_nk
count	74.000000	74.000000	74.000000	74.000000	74.000000	74.000000	74.000000	74.000000	74.000000	74.000000	74.0	74.0	74.0	74.0
mean	51.515711	-0.149984	1.283784	7.627027	0.969209	123.797297	1.013514	2.013514	273.392806	282.390473	1.0	1.0	2.0	13.0
std	0.037263	0.083297	0.510715	0.489701	0.023156	44.420463	0.561145	1.164465	136.274150	131.723345	0.0	0.0	0.0	0.0
min	51.421828	-0.479655	1.000000	6.800000	0.852511	32.000000	0.000000	1.000000	38.663651	23.235457	1.0	1.0	2.0	13.0
25%	51.497046	-0.199184	1.000000	7.225000	0.963037	91.250000	1.000000	1.000000	157.389396	163.737215	1.0	1.0	2.0	13.0
50%	51.512345	-0.135788	1.000000	7.600000	0.974736	119.500000	1.000000	2.000000	270.809020	287.811606	1.0	1.0	2.0	13.0
75%	51.541394	-0.110569	1.750000	7.900000	0.983113	159.500000	1.000000	2.000000	393.298371	402.947131	1.0	1.0	2.0	13.0
max	51.652979	0.028148	3.000000	8.900000	0.998820	225.000000	3.000000	6.000000	494.564837	498.116213	1.0	1.0	2.0	13.0

OTHER CLUSTERS OF INTEREST

Within the 14 clusters predicted, 4 clusters, besides the best pubs one, stand out.

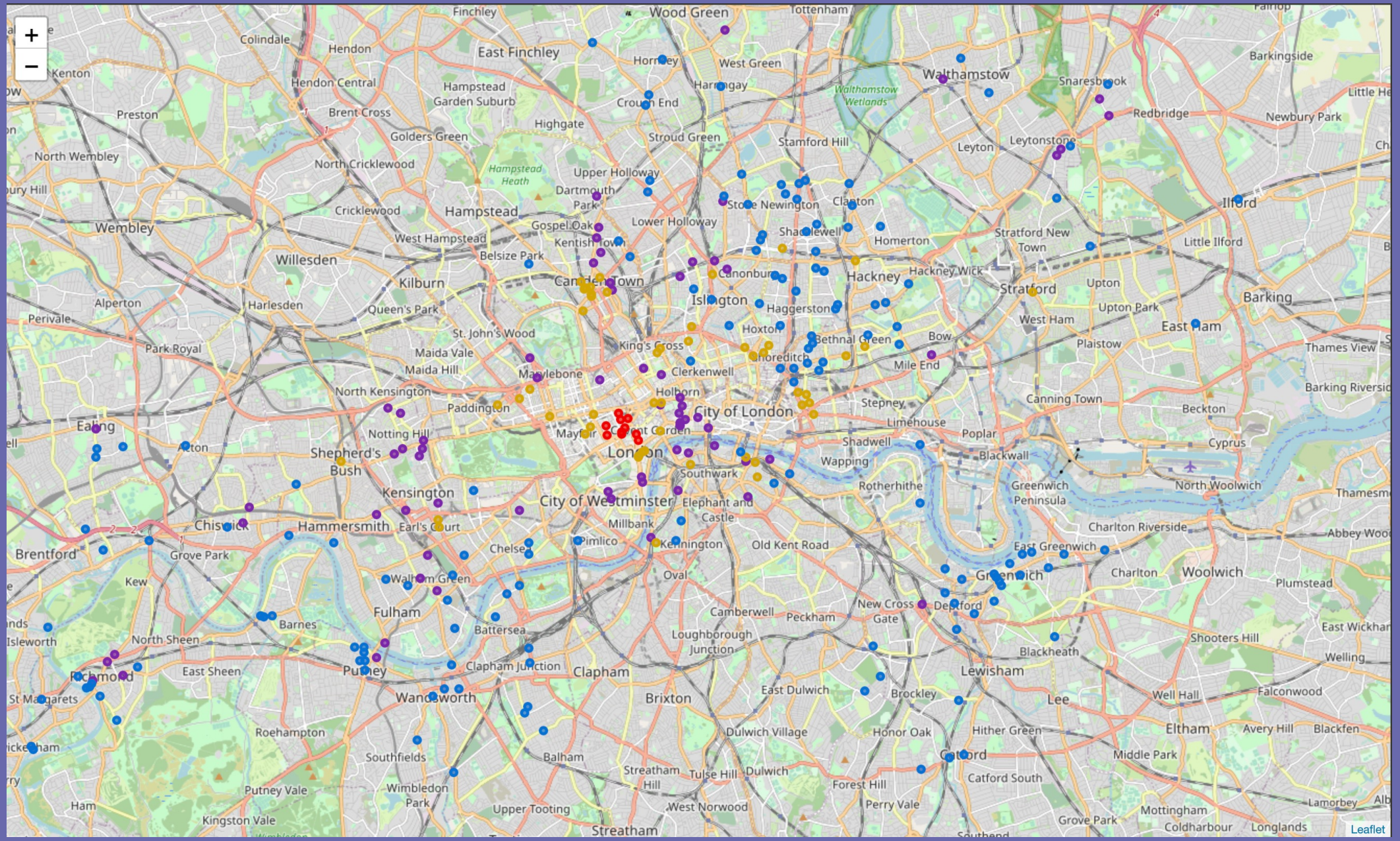
The dangerous cluster: the cluster with the highest crime rate, with a mean of 1095 crimes.

The average cluster: this cluster was in the same cluster as the best pubs when k was equal to 6. Has good rating, popularity and also close to subway and police station, but with a higher crime rate.

Good but distant clusters: two clusters belong this description. They are both good rating and popular pubs, with very low crime rate, but are distant from subway. Although one of those clusters has police station nearby.

THE PUBS MAP

- Best Pubs
- Average Pubs
- Good but distant
- Dangerous pubs

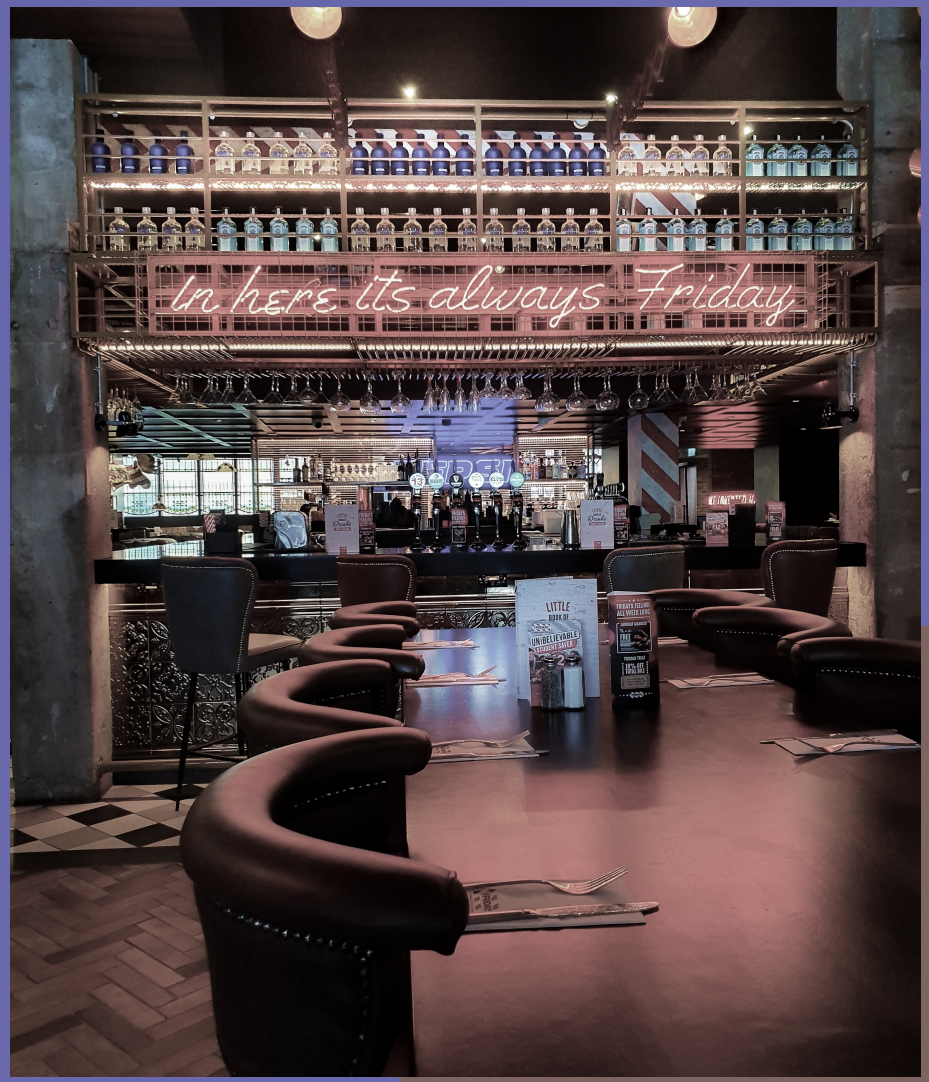


CONCLUSION

Despite the correlation between number of crimes and proximity to subway stations, the clustering model was able to find good pubs near public transportation.

Even if the best pubs did not stand out with the number of clusters suggested by the elbow method, increasing the number of k proved enough to find the best and safer pubs for women in London.

To enhance this project, data regarding tips and reviews could be gathered and analyzed to observe women points of view about a pub using Natural Language Processing.



REFERENCES

BBC (2022, January, 27). Covid: Rape reports surge after Sarah Everard murder. https://www.bbc.com/news/uk-60156459

Gilder, L. ; Clarke, J. (2022, April 5). How many violent attacks and sexual assaults on women are there? BBC News. https://www.bbc.com/news/explainers-56365412

Greater London Authority (2019). 2019 London Pubs Annual Data Note. Cultural Infraestructure Report. https://www.london.gov.uk/sites/default/files/pubs_note_final_with_cover_v2.pdf

Greater London Authority (2022). Tackling violence against women and girls. https://www.london.gov.uk/what-we-do/mayors-office-policing-and-crime-mopac/our-priorities/tackling-violence-against-women-and-girls

Leader, A. (2020).Women drive pub visit growth. The Morning Advertiser. https://www.morningadvertiser.co.uk/Article/2020/03/12/Which-social-groups-use-the-pub-most

Statista (2014). Share of adults visiting the pub weekly in the United Kingdom (UK) between 2010 and 2014, by gender. https://www.statista.com/statistics/388085/weekly-pub-visits-by-gender-adults-united-kingdom/

Thatcher, N. (2020). Fifth of women 'left out' because of gender in a pub. The Morning Advertiser. https://www.morningadvertiser.co.uk/Article/2020/02/11/How-many-women-feel-left-out-in-pubs#:~:text=The%20research%20from%20888Poker%2C%20which,fear%20of%20feeling%20left%20out.

The University Caterers Organisation (2020). Highlights from new report show increase in women drinking beer more often, growth in 'no and low alcohol' beers, and increase in independent brewery taprooms across UK. https://www.tuco.ac.uk/insight/news-opinion/highlights-new-report-show-increase-women-drinking-beer-more-often-growth-no