BOSTON'S 311

An Inside Story of 311 Service Requests

By

Bharath Kumar Shivakumar

WHAT IS THE PROJECT?

The Mayor and his Cabinet have asked the Citywide Analytics Team to look at all cases submitted to BOS: 311 in January 2020 to understand current constituent concerns and performance trends across city departments

WHAT IS THE ANALYSIS?

This is an analysis with observations and potential items for further analysis or action for the Mayor and the Analytics Team to take. Includes data visualizations that will help illustrate findings and recommendations.

- What services are people requesting?
- What do you see at a neighborhood level?
- City Operations: How long does it take to complete a request?
- What do you see at the departmental-level?
- Are there problems you see in the dataset that would pose challenges to future analyses?
- How can the City use the information in this dataset—both within City Hall and with the public—to better connect with communities?

WHERE DID THE DATA COME FROM?

Home > Organizations > Boston 311 > 311 Service Requests > 311 Service Requests - 2020

311 SERVICE REQUESTS - 2020

URL here

DATA CLEANING

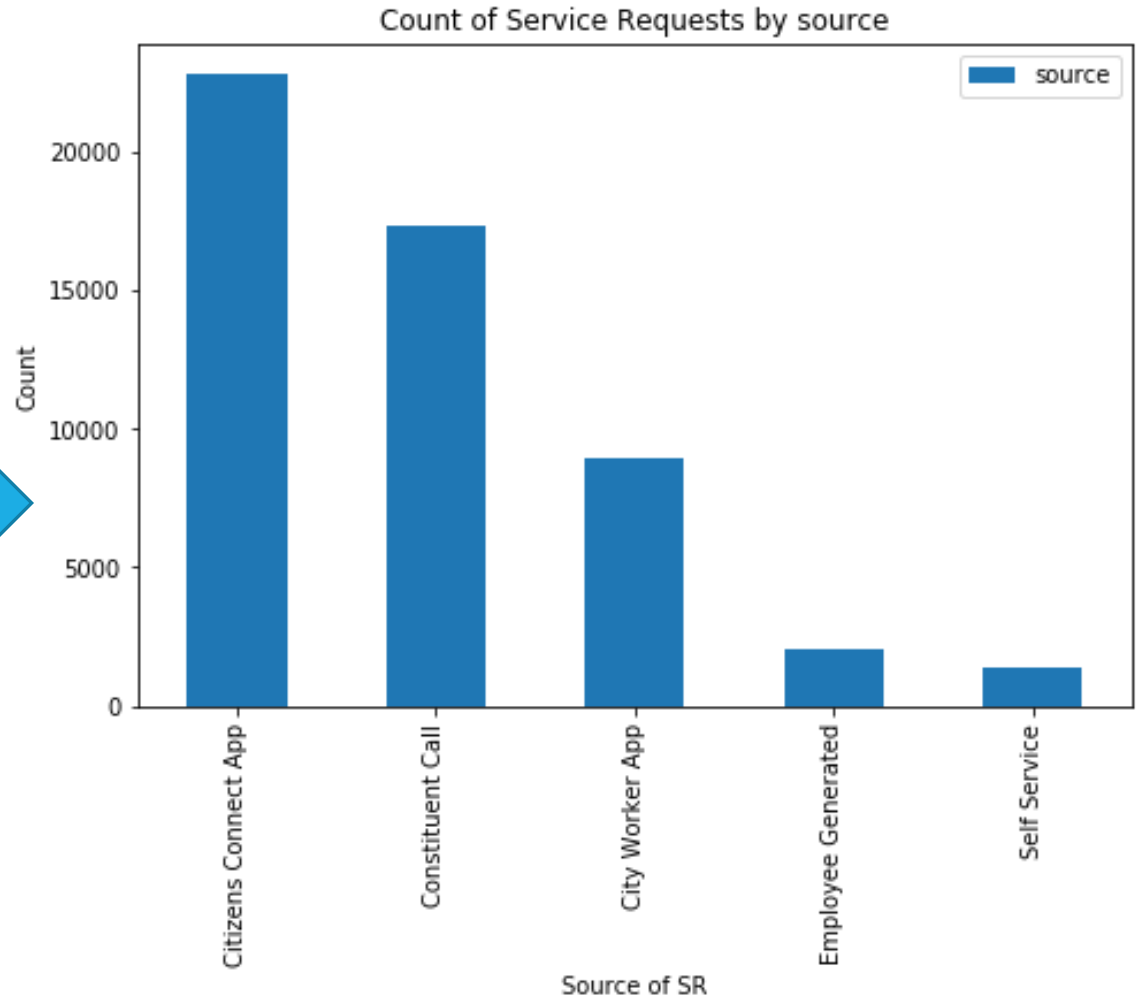
- 20 of 34 columns imported
- fillna(0, inplace = True) to replace obvious data points
- regex = True for in string replacement
- replace() to remove unnecessary trailing space and texts

Code snippet

```
#Filling the NaN values in columns submittedphoto and closedphoto with False
bdata['submittedphoto'].fillna('False', inplace = True)
bdata['closedphoto'].fillna('False', inplace = True)

#Setting the image url to true, only to get the count of submissions
bdata.closedphoto.replace('^https:.*', 'True', regex = True, inplace = True)
bdata.submittedphoto.replace('^https:.*', 'True', regex = True, inplace = True)

#Removing Ward n from the column ward
bdata.ward.replace('^Ward ', '', regex = True, inplace = True)
bdata.ward.replace(' ', 0, inplace = True)
bdata['location_zipcode'].fillna('No Zip code', inplace = True)

#Restructure neighborhood_services_district column; this will be used with apriori()
bdata.neighborhood_services_district.replace(' ', 0, inplace = True)
bdata['neighborhood_services_district'].fillna(0, inplace = True)
bdata['neighborhood_services_district'] = bdata['neighborhood_services_district'].astype('int32')
```

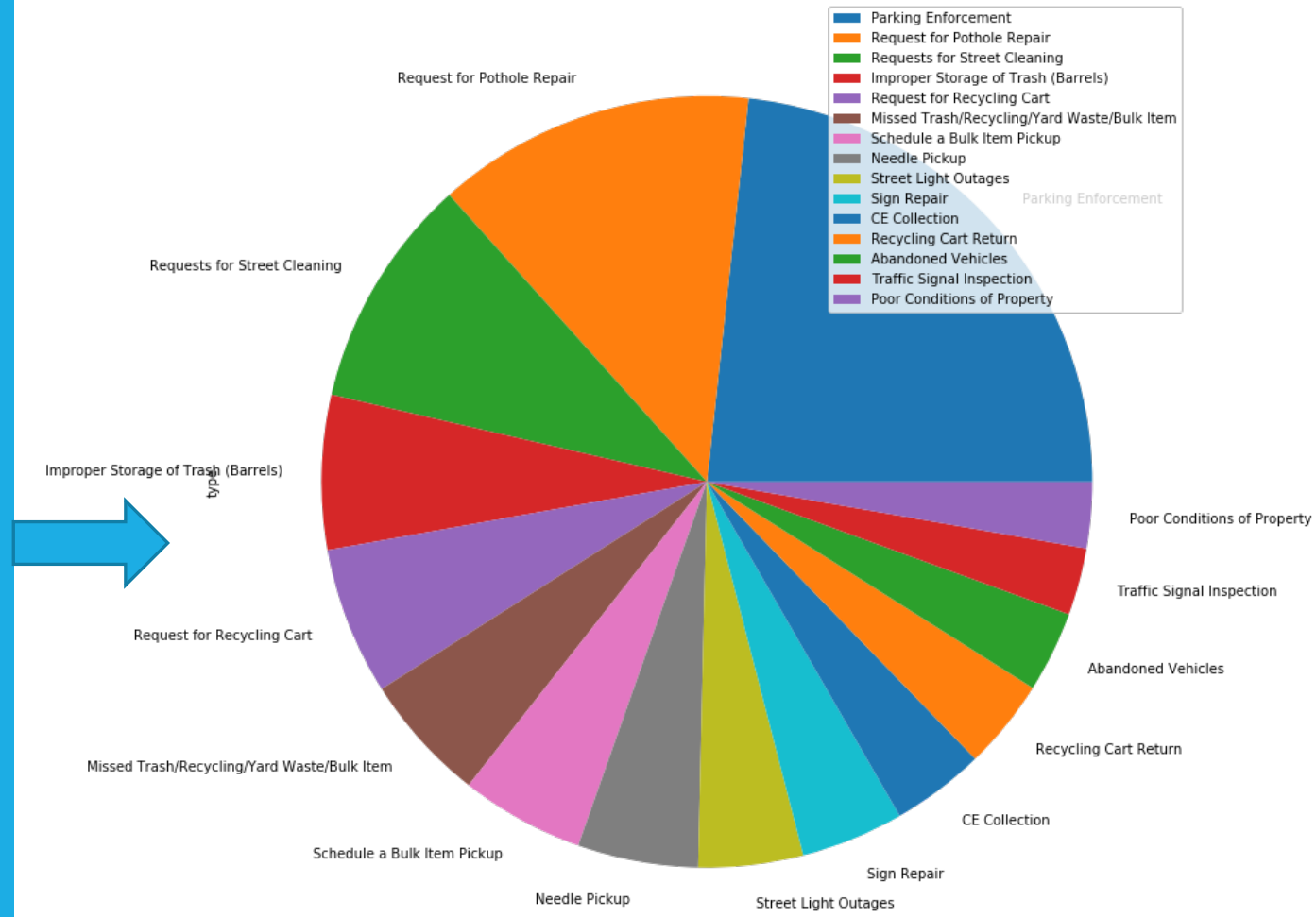
GENERAL TRENDS IN CONSTITUENT ENGAGEMENT

```
source = bdata['source'].value_counts()
source.cumsum()
source_plt = plt.figure()
source_ax = source_plt.add_axes([0,0,1,1])
source.plot(kind = 'bar')
source_ax.set_title('Count of Service Requests by source')
source_ax.set_xlabel('Source of SR')
source_ax.set_ylabel('Count')
plt.legend(loc='best')
plt.show()
```



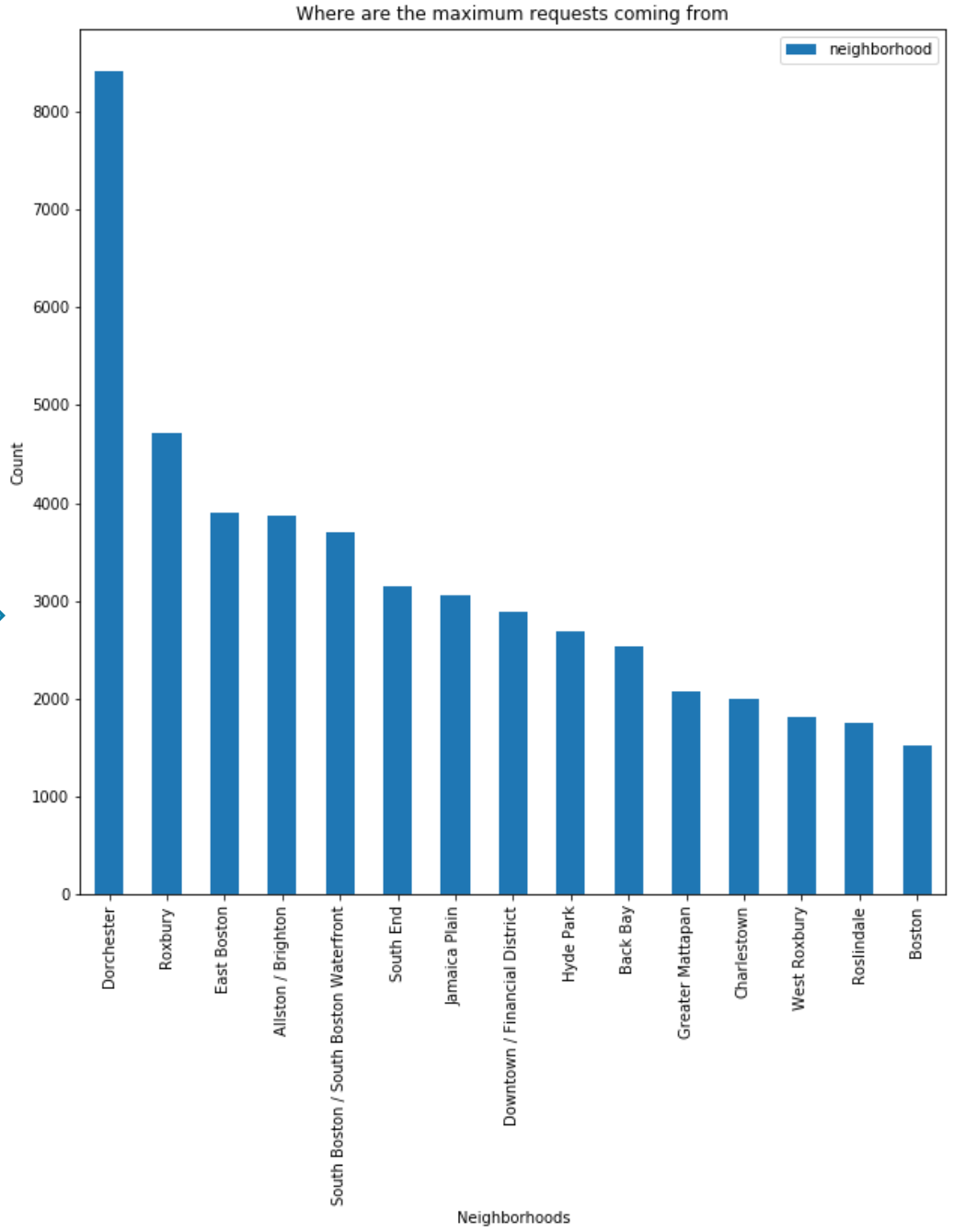
WHAT SERVICES ARE PEOPLE REQUESTING?

```
types = bdata['type'].value_counts(sort = True, dropna = True)
types = types.iloc[:15]
types_plt = plt.figure(figsize = [10,10])
types_ax = types_plt.add_axes([0,0,1,1])
types.plot(kind = 'pie')
plt.legend(loc='best')
plt.show()
print('Top 15 Request types')
types
```

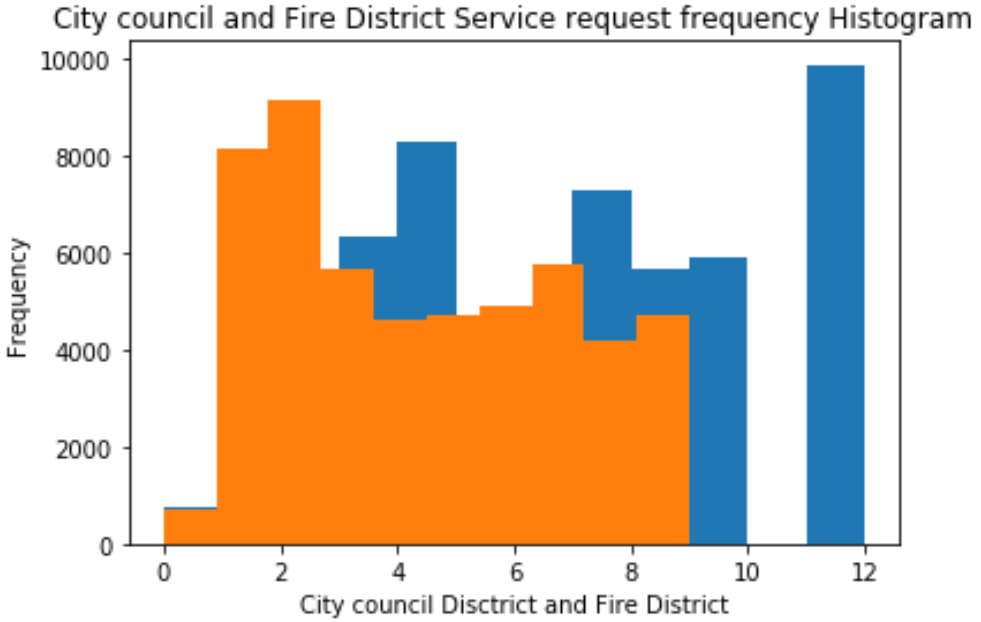


SERVICE REQUESTS AT NEIGHBORHOOD LEVEL

```
neighbor = bdata['neighborhood'].value_counts(sort = True, dropna = True)
neighbor = neighbor.iloc[:15]
neighbor_plt = plt.figure(figsize = [8,8])
neighbor_ax = neighbor_plt.add_axes([0,0,1,1])
neighbor.plot(kind = 'bar')
neighbor_ax.set_title('Where are the maximum requests coming from')
neighbor_ax.set_xlabel('Neighborhoods')
neighbor_ax.set_ylabel('Count')
plt.legend(loc='best')
plt.show()
print('Top 15 Neighborhoods')
neighbor
```



REQUEST FREQUENCY FROM CITY COUNCIL AND FIRE DISTRICT

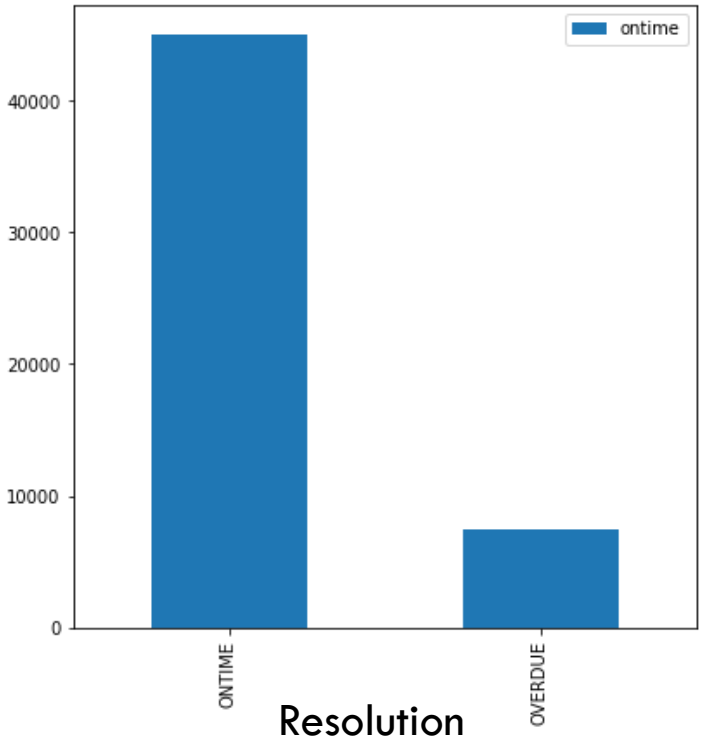



```
# Arguments required to build Histogram for fire district
bdata['fire_district'].fillna(0, inplace = True)
fire = bdata.fire_district.replace(' ', 0)
fire = fire.astype('int32')
plt.hist(fire, bins = 12)

# Arguments required to build Histogram for City council district
bdata['city_council_district'].fillna(0, inplace = True)
city = bdata.city_council_district.replace(' ', 0)
city = city.astype('int32')

# Histogram
plt.hist(city, bins = 10)
plt.xlabel('City council Disctrict and Fire District')
plt.ylabel('Frequency')
plt.title('City council and Fire District Service request frequency Histogram')
plt.show()
```

RESOLUTION, SLA BREACH AND REQUEST STATUS

- Average days to resolve a request: 2 Days and 13 Hours
- Min days required: 0 Days (Same day)
- Max days required: 75 Days

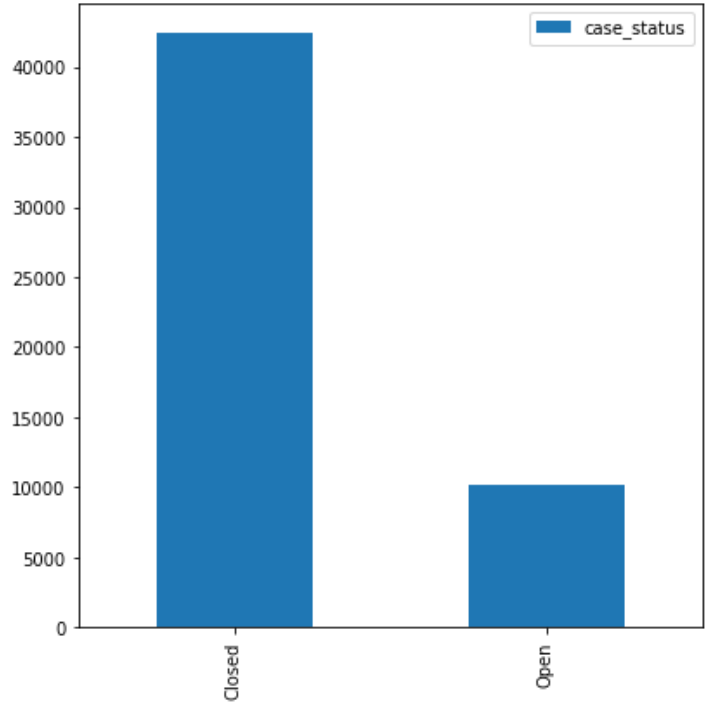
```
bdata['open_dt'] = pd.to_datetime(bdata['open_dt'])
bdata['closed_dt'] = pd.to_datetime(bdata['closed_dt'])
bdata['res_days'] = (bdata['closed_dt'] - bdata['open_dt']).dt.days
bdata['res_days'].describe()
```



Resolution

On time 45024

Overdue 7493



Service Request Status

Closed 42364

Open 10153

ASSOCIATION ANALYSIS

Why Association Analysis?

- To find common patterns of service requested by the people
- Good to know; Other applications include Online shopping recommendations
- Mathematically simple to implement and infer meaningful information from analysis output

Approach: At the neighborhood level, when grouped the data by department and reason we can see a clear correlation between the Reason for the request and issues people are facing at neighborhood level.

ASSOCIATION ANALYSIS

(LEARNING FROM WEEK 5 CLASS)

Why Association Analysis?

- To find common patterns of service requested by the people
- Good to know; Other applications include Online shopping recommendations
- Mathematically simple to implement and infer meaningful information from analysis output

Approach: At the neighborhood level, when grouped the data by department and reason we can see a clear correlation between the Reason for the request and issues people are facing at neighborhood level.

Package used: mlxtend

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

PRE-PROCESSING DATA FRAME

Restructuring the columns

```
bdata['reason'] = bdata['reason'].str.strip()
bdata['department'] = bdata['department'].astype('str')
closeCase = (bdata[bdata['case_status'] == 'Closed'].groupby(['department', 'reason'])
             ['neighborhood_services_district'].sum().unstack().reset_index().fillna(0).set_index('department'))
```

```
def cases(caseID):
    if caseID <= 0:
        return 0
    if caseID >= 1:
        return 1

closeCase_collection =  closeCase.applymap(cases)

frequent_cases = apriori(closeCase_collection, min_support = 0.01, use_colnames = True)
```

INTERPRETING RESULTS

```
frequent_cases = frequent_cases[frequent_cases['support'] >= 0.5]
frequent_cases
```

	support	itemsets
1	0.636364	(Administrative & General Requests)
7	0.545455	(Code Enforcement)
9	0.636364	(Enforcement & Abandoned Vehicles)
15	0.636364	(Highway Maintenance)
26	0.545455	(Signs & Signals)
27	0.545455	(Street Cleaning)
55	0.545455	(Enforcement & Abandoned Vehicles, Administrat...
60	0.545455	(Administrative & General Requests, Highway Ma...
70	0.545455	(Signs & Signals, Administrative & General Req...
147	0.545455	(Code Enforcement, Street Cleaning)
179	0.545455	(Enforcement & Abandoned Vehicles, Signs & Sig...
507	0.545455	(Enforcement & Abandoned Vehicles, Administrat...

Support is the relative frequency that the rules show up. In this project, I want to look for high support in order to make sure there exists a useful relationship between Reason and Neighborhood issues. And a low support is useful to find “hidden” relationships.

INTERPRETING RESULTS

Confidence is a measure of the reliability of the rule. A confidence of .5 in the above example would mean that in 50% of the cases where Street Cleaning and Code Enforcement are closely related

Lift is the ratio of the observed support to that expected if the two rules were independent. The basic rule of thumb is that a lift value close to 1 means the rules were completely independent. Lift values > 1 are generally more "interesting" and could be indicative of a useful rule pattern.

```
likely_conditions = association_rules(frequent_cases, metric = 'lift', min_threshold = 1)
likely_conditions.sort_values(by=['confidence'], ascending = False, inplace = True)
likely_conditions[(likely_conditions['lift'] >= 1) & (likely_conditions['confidence'] >= 0.7)]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
4	(Signs & Signals)	(Administrative & General Requests)	0.545455	0.636364	0.545455	1.000000	1.571429	0.198347	inf
6	(Code Enforcement)	(Street Cleaning)	0.545455	0.545455	0.545455	1.000000	1.833333	0.247934	inf
7	(Street Cleaning)	(Code Enforcement)	0.545455	0.545455	0.545455	1.000000	1.833333	0.247934	inf
9	(Signs & Signals)	(Enforcement & Abandoned Vehicles)	0.545455	0.636364	0.545455	1.000000	1.571429	0.198347	inf
10	(Enforcement & Abandoned Vehicles, Administrat...	(Signs & Signals)	0.545455	0.545455	0.545455	1.000000	1.833333	0.247934	inf