Project - 1

November 1, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
```

[2]:
```
service_311 = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv")
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3063:
DtypeWarning: Columns (48,49) have mixed types.Specify dtype option on import or
set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[3]:
```
service_311.head()
```

[3]:
```
   Unique Key            Created Date    Closed Date Agency  \
0    32310363  12/31/2015 11:59:45 PM  01-01-16 0:55   NYPD
1    32309934  12/31/2015 11:59:44 PM  01-01-16 1:26   NYPD
2    32309159  12/31/2015 11:59:29 PM  01-01-16 4:51   NYPD
3    32305098  12/31/2015 11:57:46 PM  01-01-16 7:43   NYPD
4    32306529  12/31/2015 11:56:58 PM  01-01-16 3:24   NYPD

                       Agency Name           Complaint Type  \
0  New York City Police Department  Noise - Street/Sidewalk
1  New York City Police Department         Blocked Driveway
2  New York City Police Department         Blocked Driveway
3  New York City Police Department          Illegal Parking
4  New York City Police Department          Illegal Parking

                    Descriptor    Location Type  Incident Zip  \
0             Loud Music/Party  Street/Sidewalk       10034.0
1                    No Access  Street/Sidewalk       11105.0
2                    No Access  Street/Sidewalk       10458.0
3  Commercial Overnight Parking  Street/Sidewalk       10461.0
4             Blocked Sidewalk  Street/Sidewalk       11373.0

        Incident Address  … Bridge Highway Name Bridge Highway Direction  \
0    71 VERMILYEA AVENUE  …                 NaN                      NaN
```

```
1         27-07 23 AVENUE  …                 NaN                       NaN
2  2897 VALENTINE AVENUE  …                 NaN                       NaN
3    2940 BAISLEY AVENUE  …                 NaN                       NaN
4           87-14 57 ROAD  …                 NaN                       NaN

  Road Ramp Bridge Highway Segment Garage Lot Name Ferry Direction  \
0       NaN                    NaN             NaN             NaN
1       NaN                    NaN             NaN             NaN
2       NaN                    NaN             NaN             NaN
3       NaN                    NaN             NaN             NaN
4       NaN                    NaN             NaN             NaN

  Ferry Terminal Name   Latitude  Longitude  \
0                 NaN  40.865682 -73.923501
1                 NaN  40.775945 -73.915094
2                 NaN  40.870325 -73.888525
3                 NaN  40.835994 -73.828379
4                 NaN  40.733060 -73.874170

                                   Location
0   (40.86568153633767, -73.92350095571744)
1  (40.775945312321085, -73.91509393898605)
2  (40.870324522111424, -73.88852464418646)
3   (40.83599404683083, -73.82837939584206)
4  (40.733059618956815, -73.87416975810375)

[5 rows x 53 columns]
```

[4]:
```
service_311.shape
```

[4]:
```
(300698, 53)
```

[5]:
```
import datetime
```

[6]:
```
from datetime import time
```

[9]:
```
df = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv",parse_dates =␣
↪["Created Date","Closed Date"])
```

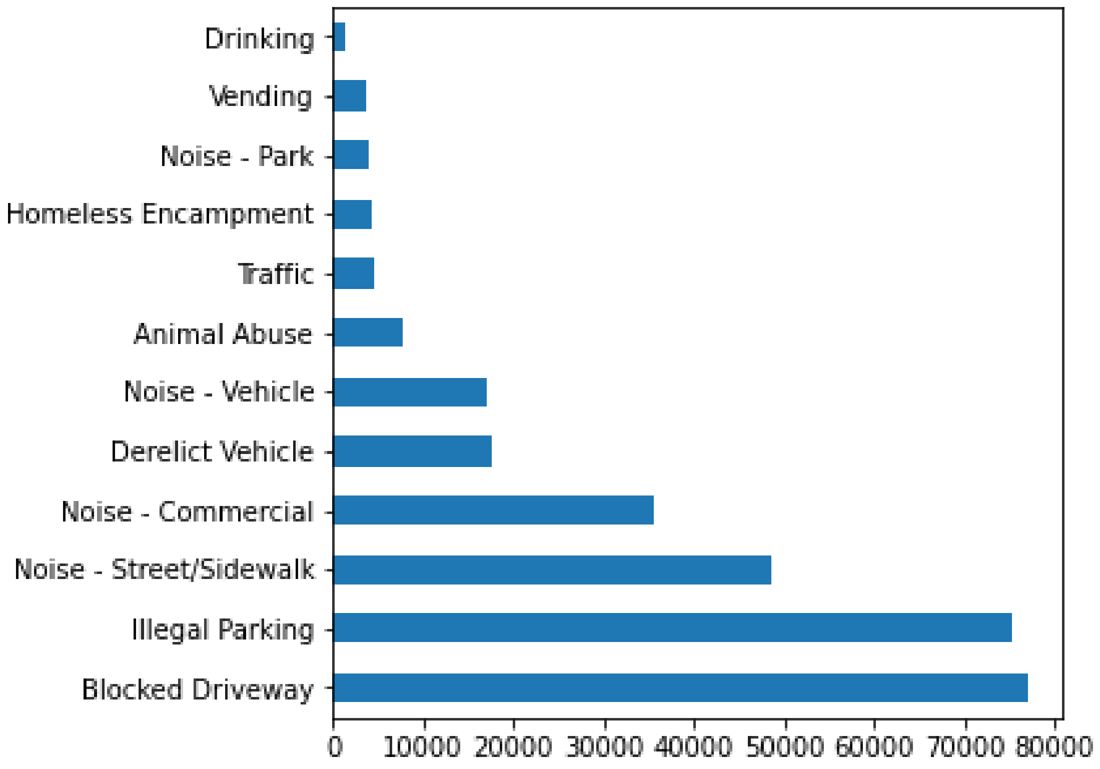
[11]:
```
df["Request_Closing_Time"] = df["Closed Date"] - df["Created Date"]
```

[12]:
```
df["Request_Closing_Time"]
```

[12]:
```
0         00:55:15
1         01:26:16
2         04:51:31
3         07:45:14
```

```
4         03:27:02
            …
300693         NaT
300694    02:00:31
300695    03:07:17
300696    04:05:33
300697    04:08:49
Name: Request_Closing_Time, Length: 300698, dtype: timedelta64[ns]
```

[15]:
```
service_311["Complaint Type"].value_counts().head(12).plot(kind =␣
↪'barh',figsize = (5,5));
```



[16]:
```
df["Request_Closing_Time"].describe()
```

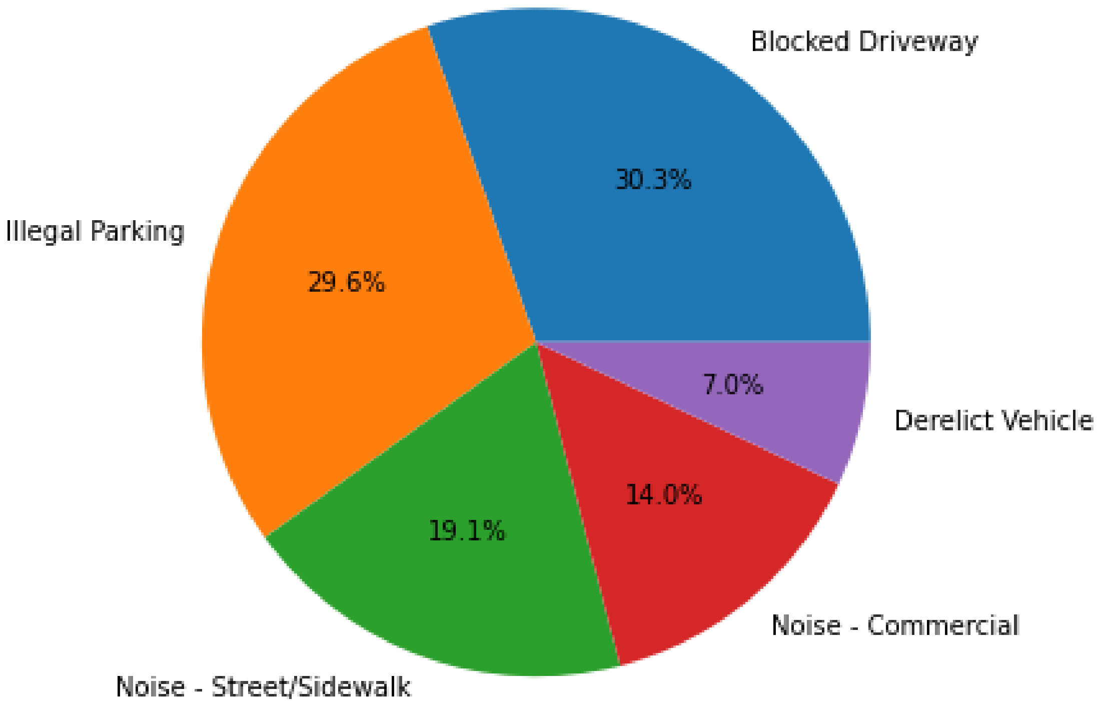
[16]:
```
count                    298534
mean      0 days 04:18:51.832782
std       0 days 06:05:22.141833
min              0 days 00:01:00
25%              0 days 01:16:33
50%       0 days 02:42:55.500000
75%              0 days 05:21:00
max             24 days 16:52:22
```

```
Name: Request_Closing_Time, dtype: object
```

[18]:
```
major_complaints = service_311.dropna(subset = ["Complaint Type"])
major_complaints = service_311.groupby("Complaint Type")
sorted_complaint_type = major_complaints.size().sort_values(ascending = False)
sorted_complaint_type = sorted_complaint_type.to_frame("count").reset_index()
sorted_complaint_type
sorted_complaint_type.head(28)
```

[18]:
```
                Complaint Type  count
0             Blocked Driveway  77044
1              Illegal Parking  75361
2      Noise - Street/Sidewalk  48612
3           Noise - Commercial  35577
4             Derelict Vehicle  17718
5              Noise - Vehicle  17083
6                 Animal Abuse   7778
7                      Traffic   4498
8          Homeless Encampment   4416
9                 Noise - Park   4042
10                     Vending   3802
11                    Drinking   1280
12    Noise - House of Worship    931
13       Posting Advertisement    650
14         Urinating in Public    592
15  Bike/Roller/Skate Chronic     427
16                 Panhandling    307
17            Disorderly Youth    286
18           Illegal Fireworks    168
19                    Graffiti    113
20               Agency Issues      6
21                    Squeegee      4
22             Ferry Complaint      2
23            Animal in a Park      1
```

[19]:
```
sorted_complaint_type = sorted_complaint_type.head()
plt.figure(figsize = (6,6))
plt.pie(sorted_complaint_type['count'],labels =␣
 ↪sorted_complaint_type["Complaint Type"],autopct = "%1.1f%%")
plt.show()
```



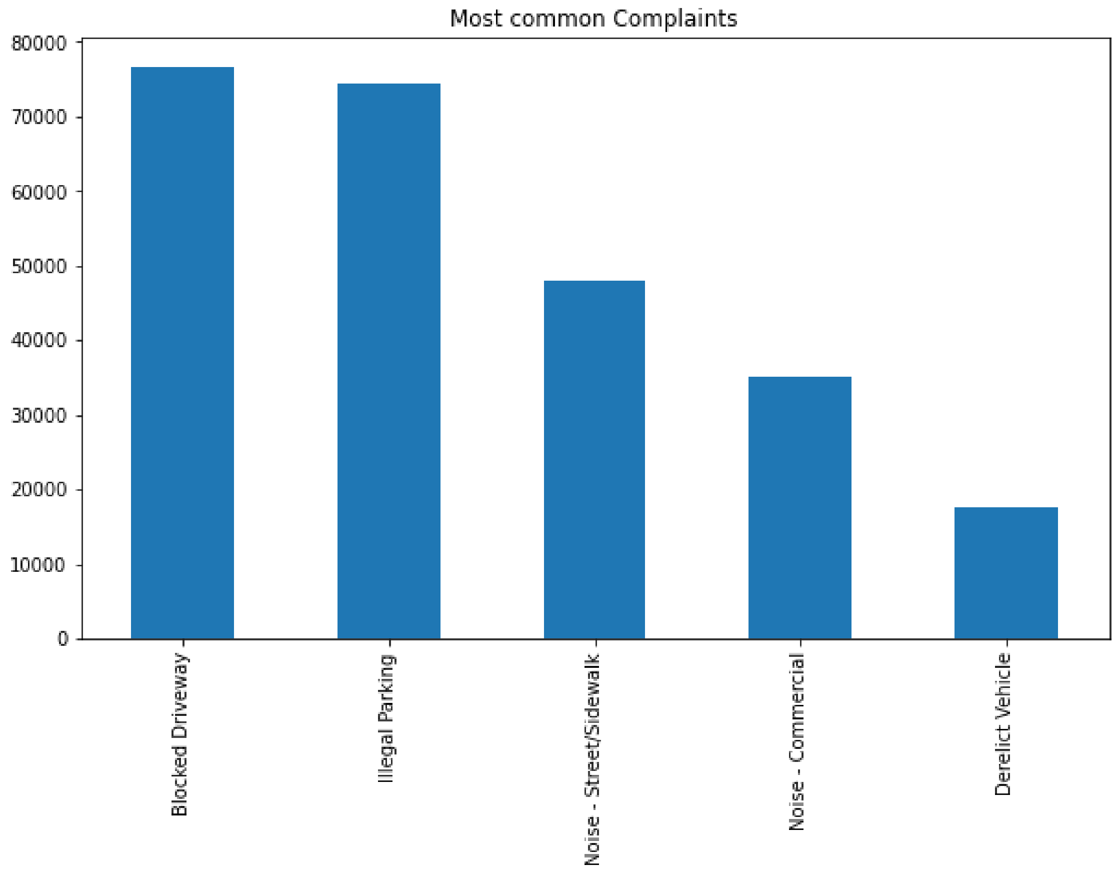
[20]:
```
def prepareData(df):
    df['Resolution_Time'] = (df['Closed Date'] - df['Created Date']).dt.
 ↪total_seconds()
    df_clean=df[df['Resolution_Time'].notnull()]
    df_perfect = df_clean[df_clean['Closed Date'] >= df_clean['Created Date']]
    df_perfect['Day of Week'] = df_perfect['Created Date'].dt.dayofweek
    df_perfect['Day of Month'] = df_perfect['Created Date'].dt.day
    df_perfect['Month'] = df_perfect['Created Date'].dt.month
    df_perfect['Year'] = df_perfect['Created Date'].dt.year
    df_perfect=df_perfect[df_perfect.Borough!='Unspecified']
    return df_perfect
```

[21]:
```
df_perfect = prepareData(df)
df_perfect.shape
```

[21]:
```
(298068, 59)
```

[22]:
```
(df_perfect['Complaint Type'].value_counts()).head().plot(kind='bar',␣
 ↪figsize=(10,6), title = 'Most common Complaints')
```

```
[22]: <AxesSubplot:title={'center':'Most common Complaints'}>
```



```
[23]: df['Request_Closing_Time'].mean()
```

```
[23]: Timedelta('0 days 04:18:51.832782')
```

```
[24]: df_avg_res_time_city = df_perfect.groupby(['City','Complaint Type']).
       ↪Resolution_Time.mean()
```

```
[26]: df_avg_res_time = df_perfect.groupby('Complaint Type').Resolution_Time.mean().
       ↪sort_values(ascending=True)
      df_avg_res_time.head(28)
```

```
[26]: Complaint Type
      Posting Advertisement          7112.891975
      Illegal Fireworks              9940.101190
      Noise - Commercial            11291.632884
      Noise - House of Worship      11495.874058
      Noise - Park                  12246.158157
      Noise - Street/Sidewalk      12377.738882
      Traffic                      12415.252002
      Disorderly Youth             12810.902098
      Noise - Vehicle              12918.914430
      Urinating in Public          13055.991554
      Bike/Roller/Skate Chronic    13523.545024
      Drinking                     13879.309748
      Vending                      14449.060358
      Squeegee                     14564.250000
      Homeless Encampment          15716.052536
      Panhandling                  15741.963934
      Illegal Parking              16149.479466
      Blocked Driveway             17057.298659
      Animal Abuse                 18768.513712
      Graffiti                     25744.504425
      Derelict Vehicle             26445.913579
      Name: Resolution_Time, dtype: float64
```

[27]:
```python
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

[28]:
```python
complaintTypecity = pd.DataFrame({'count':service_311.groupby(['Complaint␣
↪Type','City']).size()}).reset_index()
complaintTypecity
```

[28]:
```
    Complaint Type           City  count
0     Animal Abuse        ARVERNE     38
1     Animal Abuse        ASTORIA    125
2     Animal Abuse        BAYSIDE     37
3     Animal Abuse      BELLEROSE      7
4     Animal Abuse   BREEZY POINT      2
..             …              …    …
759        Vending  STATEN ISLAND     25
760        Vending      SUNNYSIDE     15
761        Vending     WHITESTONE      1
762        Vending      WOODHAVEN      6
763        Vending       WOODSIDE     15

[764 rows x 3 columns]
```

[29]:
```python
service_311.groupby(['Borough','Complaint Type','Descriptor']).size()
```

[29]:
```
Borough       Complaint Type          Descriptor
BRONX         Animal Abuse            Chained                          132
                                      In Car                            36
                                      Neglected                        673
```

```
                                    No Shelter                   71
                                    Other (complaint details)   311
                                                               …
Unspecified  Noise - Vehicle        Engine Idling                11
             Posting Advertisement  Vehicle                       1
             Traffic                Truck Route Violation         1
             Vending                In Prohibited Area            2
                                    Unlicensed                    5
Length: 288, dtype: int64
```

[30]:
```
df_dis_youth = df_perfect[df_perfect['Complaint Type']=='Disorderly Youth']
df_dis_youth = df_dis_youth.loc[:,['Resolution_Time']]
df_dis_youth.head()
```

[30]:
```
       Resolution_Time
4670             713.0
9034            4605.0
12027           2345.0
12176          19415.0
17181           6849.0
```

[31]:
```
df_noise_veh = df_perfect[df_perfect['Complaint Type']=='Noise - Vehicle']
df_noise_veh = df_noise_veh.loc[:,['Resolution_Time']]
df_noise_veh.head()
```

[31]:
```
     Resolution_Time
87           22949.0
156           7254.0
172          11319.0
221          10937.0
319           2615.0
```

[32]:
```
df_type_res = df_perfect.loc[:, ['Complaint Type','Resolution_Time']]
df_type_res.head()
df_type_res.columns
```

[32]:
```
Index(['Complaint Type', 'Resolution_Time'], dtype='object')
```

[33]:
```
fvalue, pvalue = stats.f_oneway(df_dis_youth, df_noise_veh)
pvalue
```

[33]:
```
array([0.91269878])
```

[34]:
```
df_post_ad = df_perfect[df_perfect['Complaint Type']=='Posting Advertisement']
df_post_ad = df_post_ad.loc[:,['Resolution_Time']]
df_post_ad.head()
```

[34]:
```
    Resolution_Time
39           7596.0
42           7745.0
46           7834.0
49           8042.0
51           8137.0
```

[35]:
```
df_der_veh = df_perfect[df_perfect['Complaint Type']=='Derelict Vehicle']
df_der_veh = df_der_veh.loc[:,['Resolution_Time']]
df_der_veh.head()
```

[35]:
```
     Resolution_Time
14           37763.0
151          14221.0
255           4913.0
256          14879.0
295           2712.0
```

[36]:
```
fvalue, pvalue = stats.f_oneway(df_post_ad, df_der_veh)
pvalue
```

[36]:
```
array([7.28776953e-35])
```

[37]:
```
df_perfect['Complaint_Type']=df_perfect['Complaint Type']
df_type_res = df_perfect.loc[:, ['Complaint_Type','Resolution_Time']]
model = ols('Resolution_Time ~ Complaint_Type', data=df_type_res).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

[37]:
```
                      sum_sq        df           F  PR(>F)
Complaint_Type  3.784839e+12      20.0  410.258598     0.0
Residual        1.374816e+14  298047.0         NaN     NaN
```

[38]:
```
df_city_type = pd.crosstab(df_perfect.City , df_perfect.Complaint_Type)
```

[39]:
```
from scipy.stats import chi2_contingency
from scipy.stats import chi2

table = df_city_type

stat, p, dof, expected = chi2_contingency(table)
print('dof=%d' % dof)
print(expected)

prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
```

```
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')

alpha = 1.0 - prob
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
dof=1040
[[5.73350737e+00 3.11515400e-01 5.66574169e+01 … 3.31741755e+00
  4.37007385e-01 2.80068584e+00]
 [1.64968644e+02 8.96314763e+00 1.63018841e+03 … 9.54511504e+01
  1.25738943e+01 8.05833700e+01]
 [1.86599603e+01 1.01384103e+00 1.84394139e+02 … 1.07966862e+01
  1.42226040e+00 9.11495938e+00]
 …
 [6.41892211e+01 3.48755650e+00 6.34305536e+02 … 3.71399974e+01
  4.89249632e+00 3.13549511e+01]
 [9.23615914e+01 5.01822989e+00 9.12699480e+02 … 5.34405809e+01
  7.03979170e+00 4.51165029e+01]
 [3.12736765e+00 1.69917491e-01 3.09040456e+01 … 1.80950048e+00
  2.38367665e-01 1.52764682e+00]]
probability=0.950, critical=1116.137, stat=110425.867
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)
```

[ ]:

[ ]: