Customer Service Requests Analysis

Abstract

NYC 311's mission is to provide the public with quick and easy access to all New York City government services and information while offering the best customer service. Each day, NYC311 receives thousands of requests related to several hundred types of non-emergency services, including noise complaints, plumbing issues, and illegally parked cars. These requests are received by NYC311 and forwarded to the relevant agencies such as the police, buildings, or transportation. The agency responds to the request, addresses it, and then closes it.

Presented by: Ankita Agarwal

Problem Statement:

NYC 311's mission is to provide the public with quick and easy access to all New York City government services and information while offering the best customer service. Each day, NYC311 receives thousands of requests related to several hundred types of non-emergency services, including noise complaints, plumbing issues, and illegally parked cars. These requests are received by NYC311 and forwarded to the relevant agencies such as the police, buildings, or transportation. The agency responds to the request, addresses it, and then closes it.

Problem Objective: -

Perform a service request data analysis of New York City 311 calls. You will focus on the data wrangling techniques to understand the pattern in the data and also visualize the major complaint types. Domain: Customer Service

Detailed description of the given dataset:

Field	**Description**
Unique Key	(Plain text) - Unique identifier for the complaints
Created Date	(Date and Time) - The date and time on which the complaint is raised
Closed Date	(Date and Time) - The date and time on which the complaint is closed
Agency	(Plain text) - Agency code
Agency Name	(Plain text) - Name of the agency
Complaint Type	(Plain text) - Type of the complaint
Descriptor	(Plain text) - Complaint type label (Heating - Heat, Traffic Signal Condition - Controller)
Location Type	(Plain text) - Type of the location (Residential, Restaurant, Bakery, etc)
Incident Zip	(Plain text) - Zip code for the location
Incident Address	(Plain text) - Address of the location
Street Name	(Plain text) - Name of the street
Cross Street 1	(Plain text) - Detail of cross street
Cross Street 2	(Plain text) - Detail of another cross street
Intersection Street 1	(Plain text) - Detail of intersection street if any
Intersection Street 2	(Plain text) - Detail of another intersection street if any
Address Type	(Plain text) - Categorical (Address or Intersection)
City	(Plain text) - City for the location
Landmark	(Plain text) - Empty field
Facility Type	(Plain text) - N/A

Status	(Plain text) - Categorical (Closed or Pending)
Due Date	(Date and Time) - Date and time for the pending complaints
Resolution Action Updated Date	(Date and Time) - Date and time when the resolution was provided
Community Board	(Plain text) - Categorical field (specifies the community board with its code)
Borough	(Plain text) - Categorical field (specifies the community board)
X Coordinate	(State Plane) (Number)
Y Coordinate	(State Plane) (Number)
Park Facility Name	(Plain text) - Unspecified
Park Borough	(Plain text) - Categorical (Unspecified, Queens, Brooklyn etc)
School Name	(Plain text) - Unspecified
School Number	(Plain text) - Unspecified
School Region	(Plain text) - Unspecified
School Code	(Plain text) - Unspecified
School Phone Number	(Plain text) - Unspecified
School Address	(Plain text) - Unspecified
School City	(Plain text) - Unspecified
School State	(Plain text) - Unspecified
School Zip	(Plain text) - Unspecified
School Not Found	(Plain text) - Empty Field
School or Citywide Complaint	(Plain text) - Empty Field
Vehicle Type	(Plain text) - Empty Field
Taxi Company Borough	(Plain text) - Empty Field
Taxi Pick Up Location	(Plain text) - Empty Field
Bridge Highway Name	(Plain text) - Empty Field
Bridge Highway Direction	(Plain text) - Empty Field
Road Ramp	(Plain text) - Empty Field
Bridge Highway Segment	(Plain text) - Empty Field
Garage Lot Name	(Plain text) - Empty Field
Ferry Direction	(Plain text) - Empty Field
Ferry Terminal Name	(Plain text) - Empty Field
Latitude	(Number) - Latitude of the location
Longitude	(Number) - Longitude of the location
Location	(Location) - Coordinates (Latitude, Longitude)

To Analyze:

Basic Statistics tasks: -

(Perform a service request data analysis of New York City 311 calls)

1. Import a 311 NYC service request.

2. Read or convert the columns 'Created Date' and Closed Date' to datetime datatype and create a new column 'Request_Closing_Time' as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime)

3. Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining.

4. Order the complaint types based on the average 'Request_Closing_Time', grouping them for different locations.

5. Perform a statistical test for the following:

   Please note: For the below statements you need to state the Null and Alternate and then provide a statistical test to accept or reject the Null Hypothesis along with the corresponding 'p-value'.

   a. Whether the average response time across complaint types is similar or not (overall)
   b. Are the type of complaint or service requested and location related?

Analysis and Interpretations:

1. Import a 311 NYC service request.

```
: #importing required packages
  import pandas as pd
  from pandas import Series, DataFrame
  import datetime
  import calendar
  from pylab import rcParams

  import matplotlib.pylab as plt
  %matplotlib inline
  import matplotlib
  matplotlib.style.use('ggplot')
  from matplotlib.colors import LinearSegmentedColormap

  import numpy as np

  import seaborn as sns
  sns.set(style="whitegrid", color_codes=True)

  import scipy
  plt.style.use('ggplot')
  from scipy.stats import chi2_contingency

  from statsmodels.formula.api import ols
  import statsmodels.api as sm
```

```
: data= pd.read_csv("E://Simplilearn//Data Science with Python//Projects//311-NYC
  //311_Service_Requests.csv")
```

2. Read or convert the columns 'Created Date' and Closed Date' to datetime datatype and create a new column 'Request_Closing_Time' as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime)

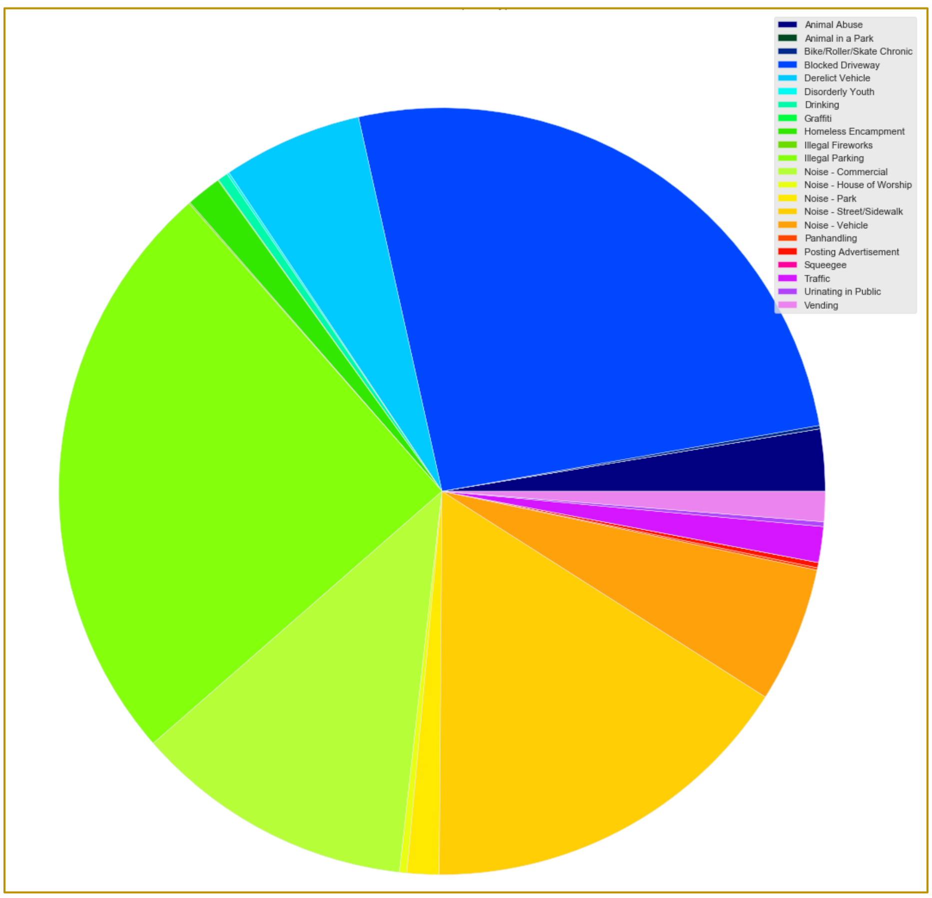
```
: data.info()

  <class 'pandas.core.frame.DataFrame'>
  Int64Index: 297965 entries, 0 to 300697
  Data columns (total 7 columns):
   #   Column          Non-Null Count   Dtype
  ---  ------          --------------   -----
   0   Unique Key      297965 non-null  int64
   1   Created Date    297965 non-null  object
   2   Closed Date     297965 non-null  object
   3   Complaint Type  297965 non-null  object
   4   Location Type   297906 non-null  object
   5   City            297965 non-null  object
   6   Borough         297965 non-null  object
  dtypes: int64(1), object(6)
  memory usage: 18.2+ MB
```

```
: data['Created Date'] = data['Created Date'].astype('datetime64[ns]')
  data['Closed Date'] = data['Closed Date'].astype('datetime64[ns]')
  data[['Created Date', 'Closed Date']].info()

  <class 'pandas.core.frame.DataFrame'>
  Int64Index: 297965 entries, 0 to 300697
  Data columns (total 2 columns):
   #   Column        Non-Null Count   Dtype
  ---  ------        --------------   -----
   0   Created Date  297965 non-null  datetime64[ns]
   1   Closed Date   297965 non-null  datetime64[ns]
  dtypes: datetime64[ns](2)
  memory usage: 6.8 MB
```

3. Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining

1 – Complaint Type Analysis – Pie Chart to show the count of various complaints.

```
#Provide major insights/patterns that you can offer in a visual format (atleast 4)

# 1. Complaint Type Analysis

freq_complaint = data.groupby('Complaint Type').agg('count')['Unique Key']

fig = plt.figure(figsize=(16,16))
ax = fig.add_subplot(111)

colormap = plt.cm.gist_ncar #nipy_spectral, Set1,Paired
colorst = [colormap(i) for i in np.linspace(0, 0.9,len(freq_complaint))]
for t,j1 in enumerate(ax.collections):
    j1.set_color(colorst[t])

labels=freq_complaint.index
plt.title('Pie Chart of Complaint type')
plt.pie(x=freq_complaint.values.astype('float64'), colors = colorst)
ax.legend(labels, loc = 'upper right')
plt.tight_layout()
plt.show()

print("STATS\n---------------------\n" , data['Complaint Type'].describe() , sep='')

print('---------------------')
print(data.groupby('Complaint Type').agg('count')['Unique Key'].sort_values())
```

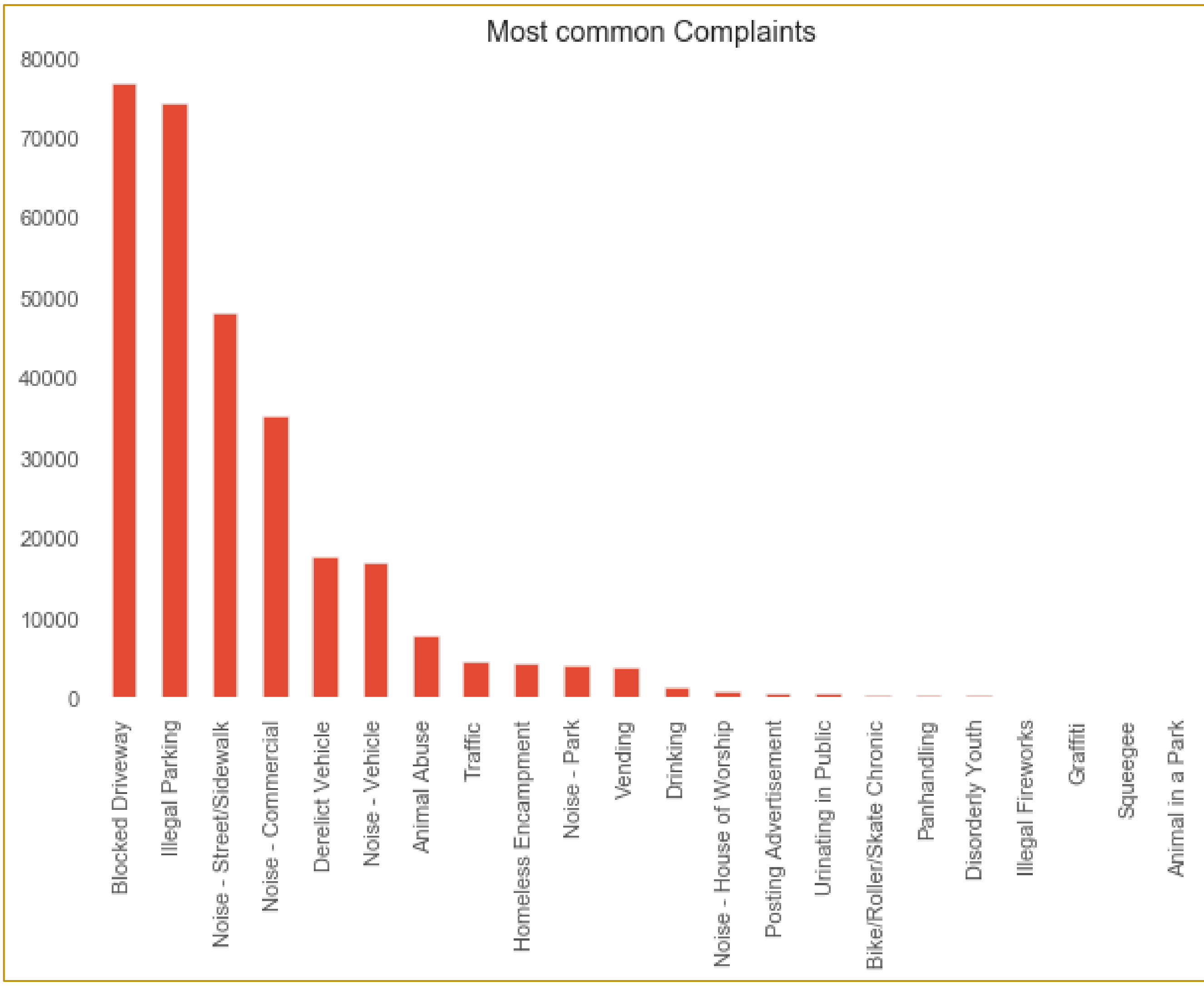


Interpretation:

It can clearly be seen that Blocked driveway (76735, blue) is the most occurring complaint in NYC followed by Illegal Parking (74294, green) whereas the least is Animal in the park (1, black).

2 – The most frequent and least frequent complaints in NYC in 2015 – represented as a bar chart.

```
# 2. the most frequent and least frequent complaints in NYC in 2015

(data['Complaint Type'].value_counts()).plot(kind='bar',
                    figsize=(10,6), title = 'Most common Complaints')

plt.box(False)
```

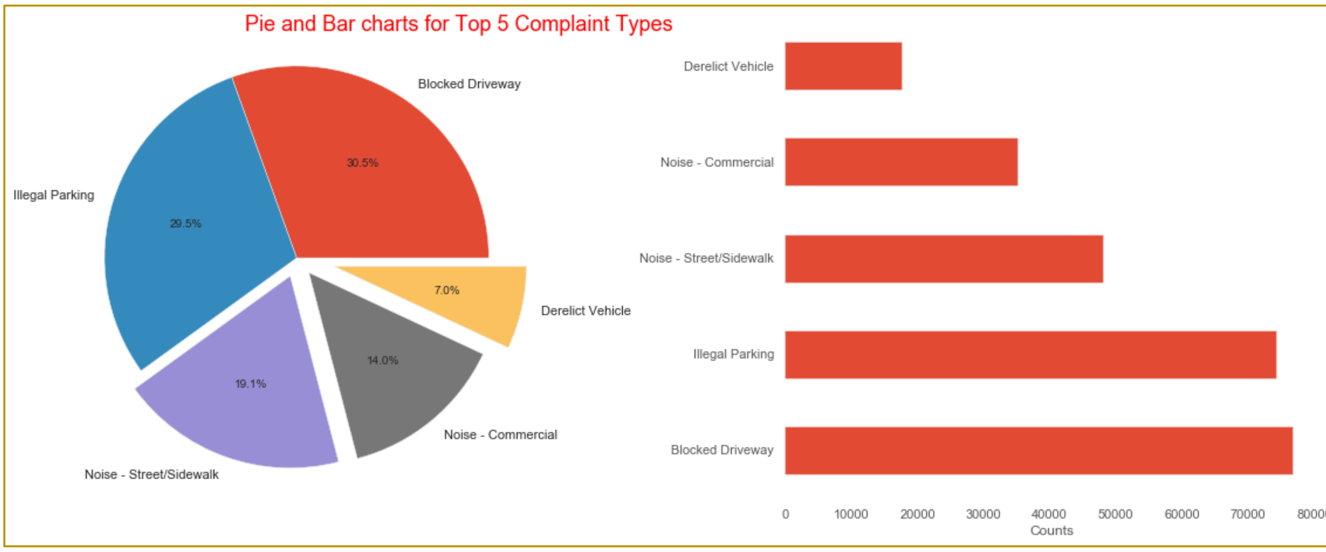


Interpretation:

It can clearly be seen that Blocked driveway is the most occurring complaint in NYC followed by Illegal Parking whereas the least is Animal in the park.

3 – Top 5 complaints types in NYC in 2015 – represented in both a pie chart and a bar chart.

```
# 3. Top 5 complaints types in NYC in 2015

most_complaint_type = pd.value_counts(data['Complaint Type'])

fig,ax = plt.subplots(1,2, figsize=[16,6.5])
fig.suptitle('Pie and Bar charts for Top 5 Complaint Types', fontsize=18, color='red', ha='right')

most_complaint_type.nlargest().plot(kind='pie',autopct='%.1f%%',ax=ax[0],explode=(0,0,0.1,0.1,0.2))
ax[0].set(ylabel='')

most_complaint_type.nlargest().plot.barh(x='Complaint Type',ax=ax[1])
ax[1].set_xlabel('Counts')
plt.tight_layout(1.2)

plt.box(False)
```

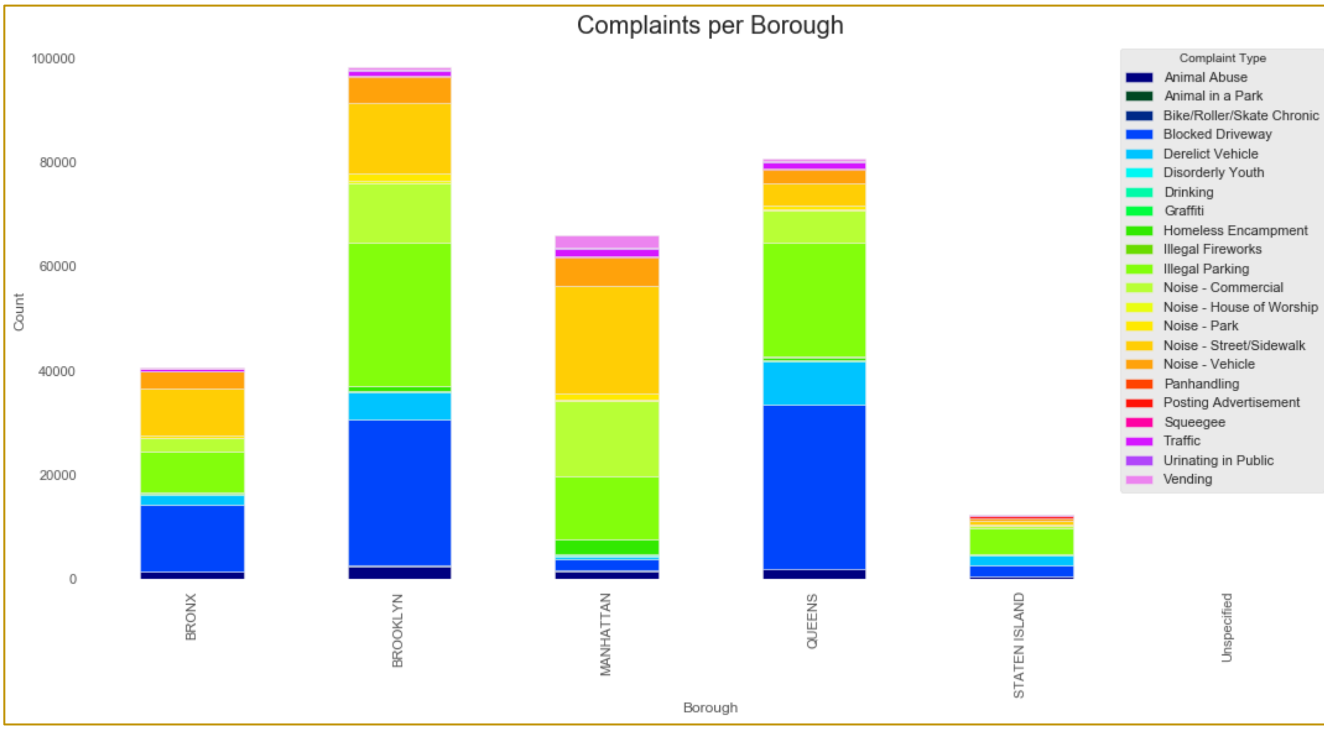


Interpretation:

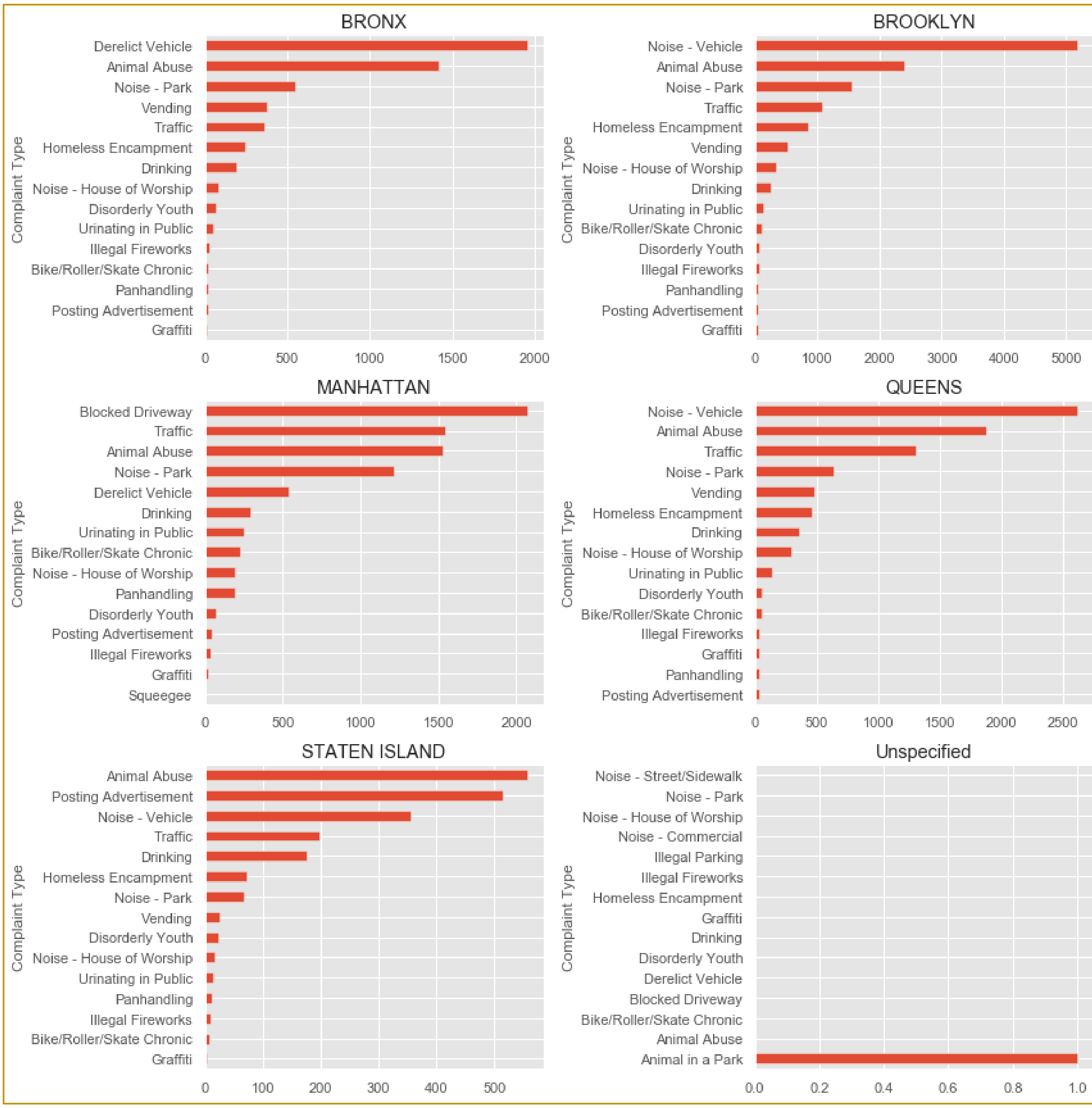
The top 5 complaints are Blocked driveway (30.5%) followed by Illegal Parking (29.5%), which constitutes 60% of all the complaints in NYC in 2015.

4 – Display the complaint type and Borough together – represented in stacked bar chart.

```
#4. Display the complaint type and Borough together

mplaintTypeBorough =data.groupby(['Borough','Complaint Type']).size()

clarity_color_table = pd.crosstab(index=data["Borough"],
                                  columns=data["Complaint Type"])

cmap1 = LinearSegmentedColormap.from_list("my_colormap", colorst)

clarity_color_table.plot(kind="bar", figsize=(18,8),stacked=True, colormap=cmap1)
plt.title('Complaints per Borough',fontsize=20)
plt.ylabel('Count')
plt.box(False)
plt.show()
```




```
# Visualization of most Complaints per Borough
borough_comp = data.groupby(['Complaint Type','Borough']).size().unstack()

col_number = 2
row_number = 3

fig, axes = plt.subplots(row_number,col_number, figsize=(12,12))

for i, (label,col) in enumerate(borough_comp.iteritems()):
    ax = axes[int(i/col_number), i%col_number]
    col = col.sort_values(ascending=True)[:15]
    col.plot(kind='barh', ax=ax)
    ax.set_title(label)

plt.tight_layout()
```

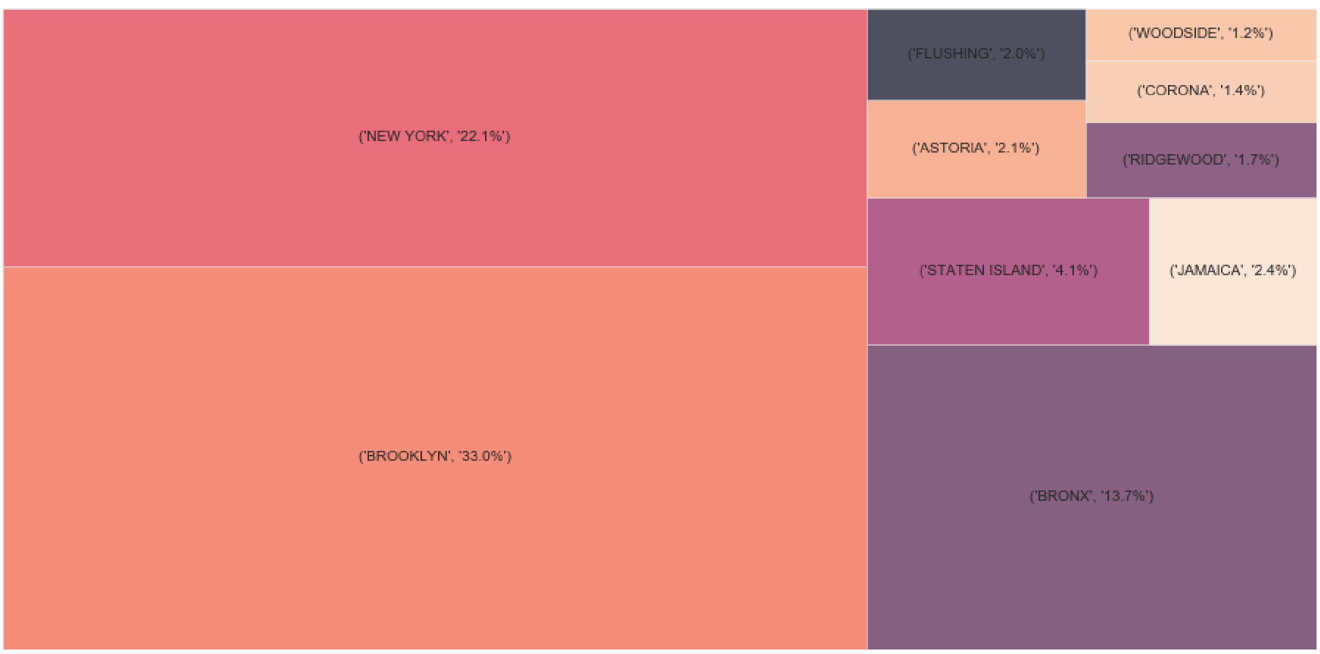


Interpretation:

Things that can be highlighted here is that Brooklyn has the maximum number of complaints in the data given with maximum complaints being Illegal Parking. Manhattan has more Noise related complaints as compared to other Boroughs. The least complaints are from Staten Island.

5 – Citywise Complaints Analysis - Top 10 – represented as a Heat Map.

```
#5. Citywise Complaints Analysis - Top 10

citywise_complaints = pd.DataFrame(data['City'].value_counts()[:10])
percent100 = data['City'].value_counts(normalize=True).mul(100).round(1).astype(str) + '%'

#pip install squarify
import squarify

fig = plt.gcf()
fig.set_size_inches(16, 8)
label=zip(list(citywise_complaints.index),percent100)
squarify.plot(sizes=citywise_complaints['City'], label=label, alpha=0.7)
plt.axis('off')
plt.show()
```

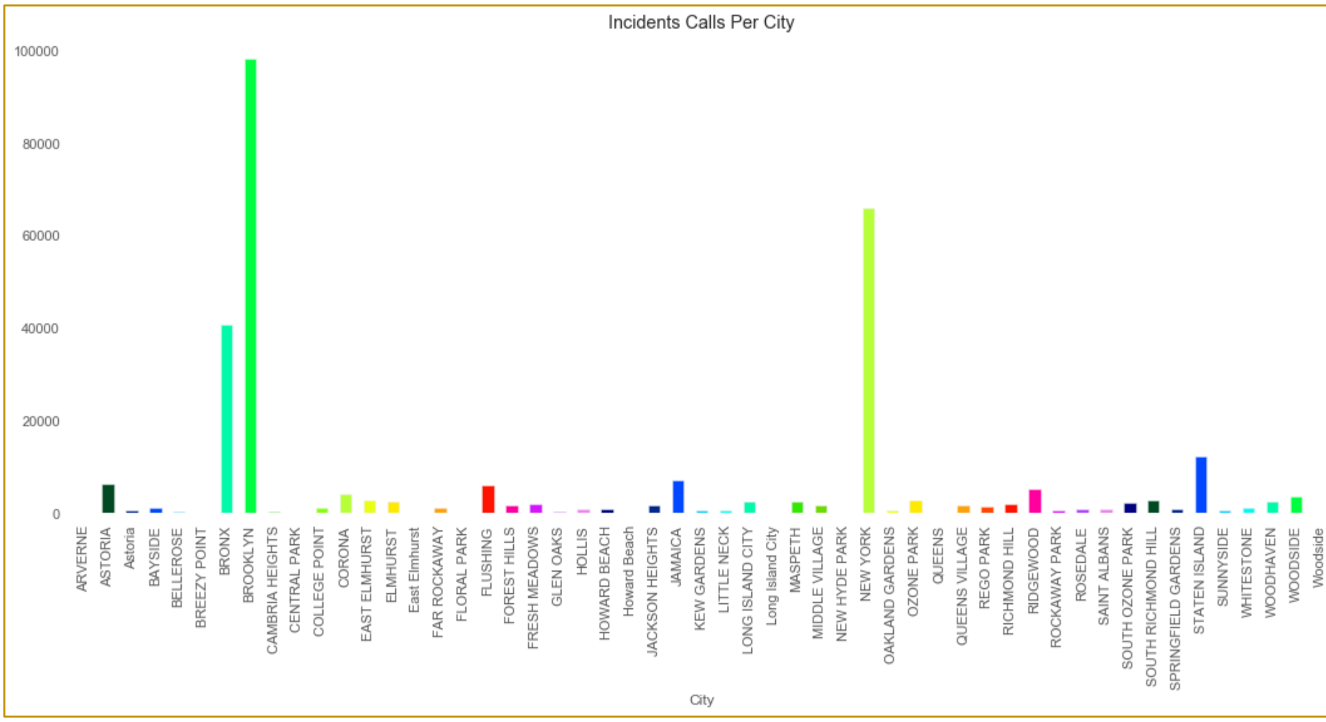


Interpretation:

The top 5 complaints are from Brooklyn (33%), New York (22.1%), Bronx (13.7%), Staten Island (4.1%) and Jamaica (2.4%).

6 – Incidents Calls Per City – represented in a bar chart.

```
#6. Incidents Calls Per City

data.groupby('City').size().plot(kind='bar', color = colorst, figsize=(18,7), title=('Incidents Calls Per City'))
plt.box(False)
```

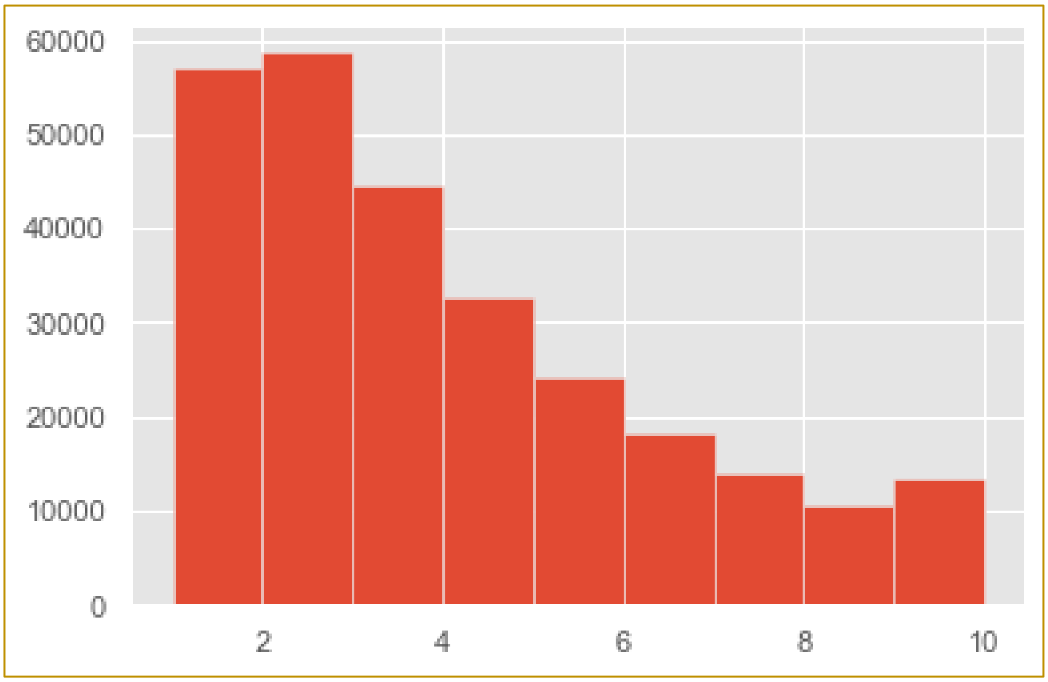


Interpretation:

The top 5 complaints are from Brooklyn, New York, Bronx, Staten Island and Jamaica.

7 – Complaint types based on the average 'Request_Closing_Time'– represented in a histogram.

```
#complaint types based on the average 'Request_Closing_Time'

data['RequestClosingHours'] = data['RequestClosingTime'].astype('timedelta64[h]')+1

mean = data['RequestClosingHours'].mean()
std = data['RequestClosingHours'].std()

dataplot = data[ ((data['RequestClosingHours']-mean)/std) < 1]
dataplot['RequestClosingHours'].hist(bins=9)
```

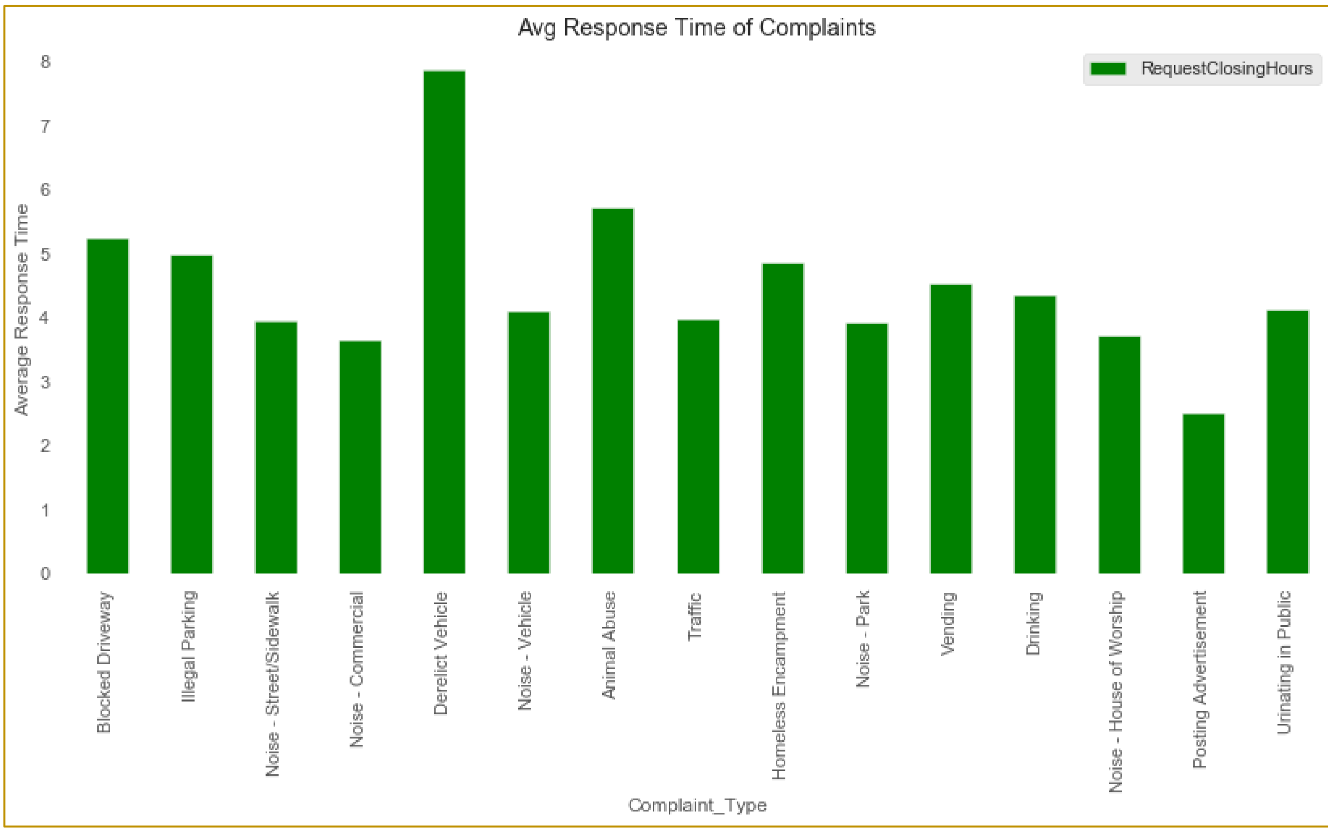


Interpretation:

Max incidents are processed between at an average of 2-3 hours i.e., The time between the Incident being reported and it being closed is at an average of 2-3 hours.

8 – Average Response Time of Complaints– represented in a bar chart.

```
#8.Average Response Time of Complaints

import matplotlib.ticker as ticker

var = data[['RequestClosingHours', 'Complaint Type']].groupby('Complaint Type').mean()
frequent = data['Complaint Type'].value_counts()

var = var.loc[frequent.index]

var.head(15).plot(kind='bar', figsize=(14,6), color = 'green')

plt.xlabel('Complaint_Type')
plt.ylabel('Average Response Time')
plt.title("Avg Response Time of Complaints")
tick_spacing = 2
ax.yaxis.set_major_locator(ticker.MultipleLocator(tick_spacing))
plt.box(False)
```



Interpretation:

The average processing time for Derelict Vehicle is the highest whereas for Posting Advertisement is the lowest. However, in major incidents the processing time seems to be an average of 4 hours.

4. Order the complaint types based on the average 'Request_Closing_Time', grouping them for different locations.

```
#complaint types based on the average 'Request_Closing_Time' based on complaint type grouped under different location

data['RequestClosingTime1'] = data['RequestClosingTime'].values.astype(np.int64)

newss = data.groupby(['Location Type' , 'Complaint Type']).mean()

newss.RequestClosingTime1.apply(pd.to_timedelta)
```

```
Location Type        Complaint Type
Bridge               Homeless Encampment   03:49:09.500000
Club/Bar/Restaurant  Drinking              04:32:44.923287
                     Noise - Commercial    03:03:43.846574
                     Urinating in Public          07:55:12
Commercial           Animal Abuse          05:20:33.967741
                                                ...
Street/Sidewalk      Urinating in Public   03:17:06.835443
                     Vending               04:01:34.806483
Subway Station       Animal Abuse          03:02:08.181818
                     Urinating in Public   01:09:07.666666
Vacant Lot           Derelict Vehicle      07:28:26.129870
Name: RequestClosingTime1, Length: 69, dtype: timedelta64[ns]
```

Interpretation:

Different locations have different types of complaints and varying closing time.

5. Perform a statistical test for the following:

Please note: For the below statements you need to state the Null and Alternate and then provide a statistical test to accept or reject the Null Hypothesis along with the corresponding 'p-value'.

a. Whether the average response time across complaint types is similar or not (overall)

Here we do and Anova test to test the relation between the Average response time and complaint type. Hypothesis Statement is as below: -

Ho: The average response time across complaint types is not similar.
Ha: The average response time across complaint types is similar.

```
#statistical test - Hypothesis Testing

#1. Whether the average response time across complaint types is similar or not (overall) - ANOVA

#Ho: The average response time across complaint types is not similar.
#Ha: The average response time across complaint types is similar.

data.columns = data.columns.str.replace('Complaint Type','Complaint_Type')
data.columns = data.columns.str.replace('Location Type','Location_Type')

mod = ols('RequestClosingTime1 ~ Complaint_Type', data = data).fit()
print(sm.stats.anova_lm(mod))
```

```
                      df        sum_sq       mean_sq           F  PR(>F)
Complaint_Type      21.0  5.214914e+30  2.483292e+29  538.297063     0.0
Residual        297943.0  1.374482e+32  4.613238e+26         NaN     NaN
```

Interpretation:

As we know, p-value < alpha, p-value is less than alpha; we reject the null hypothesis. We take alpha value as 0.05 at 95% confidence level.

We can clearly see that p-value = 0.0 signifying that these variables are not significant to each other and hence we reject the null hypothesis.

b. Are the type of complaint or service requested and location related?

Here we do and Chi-Square test to test the relation between service requested and location.

Ho: The type of complaint or service requested and location are not related.
Ha: The type of complaint or service requested and location are related.

```
#2.Are the type of complaint or service requested and location related? - CHI-SQUARE

#Ho: The type of complaint or service requested and location are not related.
#Ha: The type of complaint or service requested and location are related.

contingency_table = pd.crosstab(data['Location_Type'], data['Complaint_Type'])
chisq_statistic, p_value, ddof, expected = chi2_contingency(contingency_table.values)
print('Chi square statistic: {}, p-value: {}'.format(chisq_statistic,p_value))
```

```
Chi square statistic: 1325898.8841401357, p-value: 0.0
```

Interpretation:

As we know, p-value < alpha, p-value is less than alpha; we reject the null hypothesis. We take alpha value as 0.05 at 95% confidence level.

We can clearly see that p-value = 0.0 signifying that these variables are not significant to each other and hence we reject the null hypothesis.

Programming Codes:

Adobe Acrobat Document

---The End---