WQD7003_Group_Assingment

by Gunasegarran Magadevan

WQD7003 DATA ANALYTICS

Group Member :

¶

- Gunasegarran Magadevan - WQD170002

- Mathavan Chandrasegaram - WQD170075

Topic : Analysis and Prediction of Crime Statistic in London.

Source 1 : www.kaggle.com/jboysen/london-crime (http://www.kaggle.com/jboysen/london-crime) **- Dataset of London Crime By Borough/District**

Source 2 : https://data.gov.uk/dataset/a76f46f9-c10b-4fe7-82f6-aa928471fcd1/land-area-and-population-density-ward-and-borough (https://data.gov.uk/dataset/a76f46f9-c10b-4fe7-82f6-aa928471fcd1/land-area-and-population-density-ward-and-borough) **- Dataset of London Borough/District Dataset**

Source 3 : https://www.kaggle.com/csobral/london-borough-and-ward-boundaries-up-to-2014/version/1 (https://www.kaggle.com/csobral/london-borough-and-ward-boundaries-up-to-2014/version/1) **- Dataset of London Borough/District Map**

1. Installing & Importing Libraries

In [1]:
```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib as mpl
from scipy import stats
import geopandas as gpd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import ipdb
from collections import defaultdict
import descartes


from dtaidistance import clustering, dtw
from scipy.cluster.hierarchy import dendrogram, linkage, set_link_c
from sklearn.cluster import DBSCAN, KMeans
from sklearn.metrics import silhouette_score
from pandas.plotting import scatter_matrix
from sklearn.cluster import KMeans
```

2. Loading the Dataset

In [2]:
```
# Dataset of London Crime By Borough/District - www.kaggle.com/jboy
ds = pd.read_csv('../data/london_crime_by_lsoa.csv', dtype={'month'
```

3. Data Insight

In [3]:
```
print("Counts of data : ",ds.count())
print("\n\n")
print("Top 5 data  : ",ds.head())
```

```
Counts of data :  lsoa_code         13490604
borough           13490604
major_category    13490604
minor_category    13490604
value             13490604
year              13490604
month             13490604
dtype: int64



Top 5 data  :      lsoa_code     borough              major_catego
ry  \
0  E01001116     Croydon                    Burglary
1  E01001646   Greenwich  Violence Against the Person
2  E01000677     Bromley  Violence Against the Person
3  E01003774   Redbridge                    Burglary
4  E01004563  Wandsworth                     Robbery

               minor_category  value  year month
0  Burglary in Other Buildings      0  2016    11
1               Other violence      0  2016    11
2               Other violence      0  2015     5
3  Burglary in Other Buildings      0  2016     3
4            Personal Property      0  2008     6
```

Summary of Data Insight

The data above represented by the london_crime_by_lsoa.csv, covers the count of criminal reports by month, Lower Super Output Area (LSOA) borough (district), and major_category and minor_category from January 2008 to December 2016 in London (United Kindom) by providing 13,490,604 samples with 7 variables each.

The variables details are the followings:

- lsoa_code: LSOA in London (United Kingdom)
- borough: borough (district) names of in London (United Kingdom)
- major_category: categorization of high level crime
- minor_category: categorization of low level crime
- value: monthly reported count of categorical crime in given borough (district)
- year: year of reported counts, 2008-2016
- month: month of reported counts, 1-12 (January-December)

4. Identifying Dirty Data

In [4]:
```
ds.isnull().values.any()
```
Out[4]:
```
False
```

Summary of Identifying Dirty Data

*Used the fastest way to identify determine if **ANY** value in a series is missing.* Therefore the finding is the data are clean.

5. NUMERICAL QUANTITATIVE ANALYSIS

In [5]:
```
num_summary = ds.describe(include=np.number)
```
In [6]:
```
print('MIN: {}, MAX: {}, UNIQUE VALUES: {}, MODE: {}'.
      format(int(num_summary['value']['min']),
             int(num_summary['value']['max']),
             ds['value'].unique().shape[0],
             stats.mode(ds['value'])[0][0]))
```
```
MIN: 0, MAX: 309, UNIQUE VALUES: 247, MODE: 0
```

Summary of QUANTITATIVE VARIABLE ANALYSIS

1. Since **247** unique values of the dataset's samples have the variable **value** equals to **0**.
2. To conclude, the window of time from **2008** to **2016** was not too compact of criminal activities.

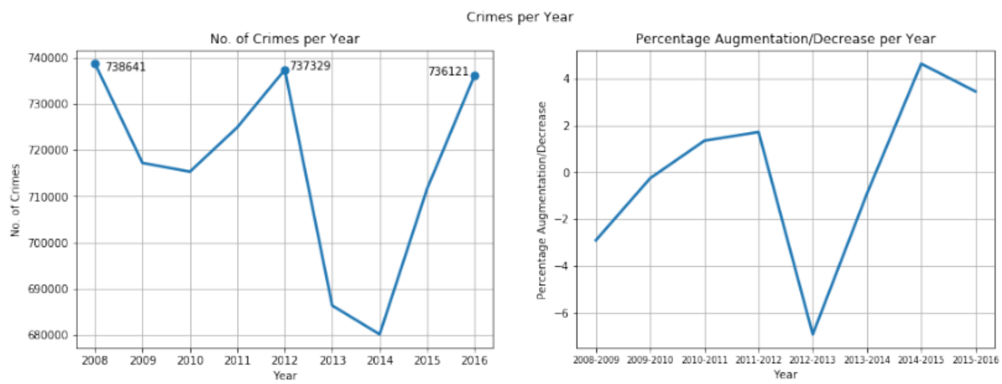
11

5.1 Crimes Per Year & Crimes Per Month

In [7]:

```
crimes_per_year, crimes_per_month = {}, {}
# pre assigning month of reported counts, 1-12 (January-December)
months = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11',
aug_dim_perc = []


for year in np.sort(ds['year'].unique()):
    crimes_y = ds.loc[(ds['year'] == year)]
    crimes_per_year[year] = sum(crimes_y['value'])
    crimes_per_month[year] = []


    for month in months:
        crimes = crimes_y.loc[crimes_y['month'] == month]
        crimes_per_month[year].append(sum(crimes['value']))

sorted_vals = list(crimes_per_year.values())
sorted_vals.sort()

cpm_matrix = np.mean([crimes_per_month[key] for key in crimes_per_m
                     axis=0)

for year in list(crimes_per_year.keys())[1:]:
    pr_year = str(int(year) - 1)
    score = ((crimes_per_year[year] / crimes_per_year[pr_year]) * 1

    aug_dim_perc.append(round(score, 2))
```

In [8]:
```
plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.plot(range(len(crimes_per_year.keys())), list(crimes_per_year.va
         linewidth=2.5, marker='o', markersize=7.0,
         markevery=[0, 4, 8])
plt.annotate(sorted_vals[-1], (0.2, sorted_vals[-1] - 1500))
plt.annotate(sorted_vals[-2], (4.1, sorted_vals[-2]))
plt.annotate(sorted_vals[-3], (7., sorted_vals[-3]))
plt.xticks(range(len(crimes_per_year.keys())), crimes_per_year.keys
plt.xlabel('Year')
plt.ylabel('No. of Crimes')
plt.title('No. of Crimes per Year')
plt.grid()

plt.subplot(1, 2, 2)
plt.plot(range(len(aug_dim_perc)), aug_dim_perc, linewidth=2.5)
plt.grid()
plt.xticks(ticks=range(len(aug_dim_perc)),
           labels=['2008-2009', '2009-2010', '2010-2011', '2011-201
                   '2012-2013', '2013-2014', '2014-2015', '2015-201
           fontsize=8)
plt.xlabel('Year')
plt.ylabel('Percentage Augmentation/Decrease')
plt.title('Percentage Augmentation/Decrease per Year')

plt.suptitle("Crimes per Year")
#plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.savefig('../images/crimes_per_year.pdf', bbox_inches='tight')
```



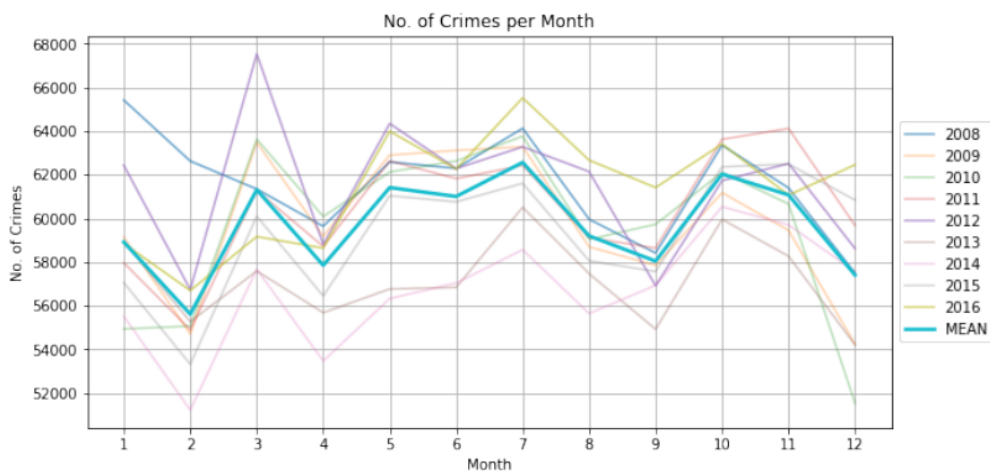
Summary of Crimes Per Year

The figure above represents the flow of criminal activities on a by yearly basis:

1. The most criminally decrease year are by year **2008**, **2012** and **2016**.
2. The most peaceful year are **2014** and **2013**.

28

In [9]:
```
plt.figure(figsize=(10, 5))

for year in crimes_per_month.keys():
    if int(year) in [2008, 2012, 2016]:
        plt.plot(range(len(crimes_per_month[year])), crimes_per_mon
                 label=year, alpha=.6)
    else:
        plt.plot(range(len(crimes_per_month[year])), crimes_per_mon
                 label=year, alpha=.3)
plt.plot(range(12), cpm_matrix, label='MEAN', linewidth=2.5)

plt.xticks(range(len(ds['month'].unique())), months)
plt.xlabel('Month')
plt.ylabel('No. of Crimes')
plt.title('No. of Crimes per Month')
plt.grid()
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))

plt.savefig('../images/crimes_per_month.pdf',
            bbox_inches='tight')
```



Summary of Crimes Per Month

The figure above represents the flow of criminal activities on a by month basis:

1. Observing a behaviour that remains coherent with the flow of criminal activities on a by yearly basis.

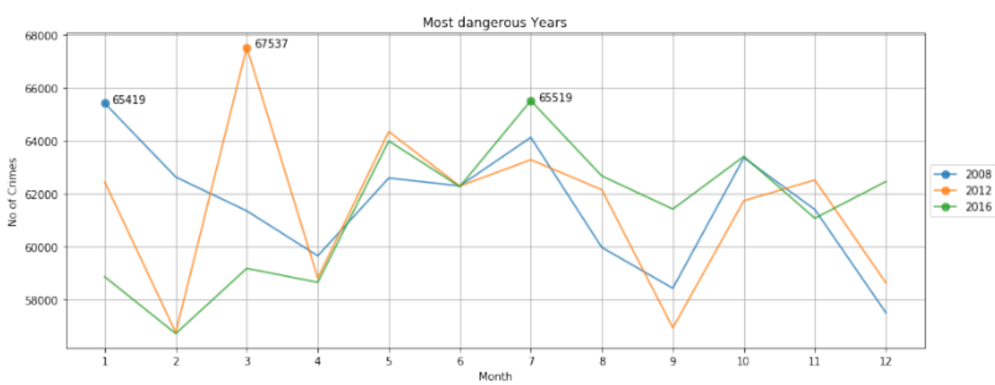
6.2 Most Dangerous Years

In [10]:

```python
crimes_per_year, crimes_per_month = {}, {}

for year in ['2008', '2012', '2016']:
    crimes_y = ds.loc[(ds['year'] == year)]
    crimes_per_month[year] = []

    for month in ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'
        crimes = crimes_y.loc[crimes_y['month'] == month]
        crimes_per_month[year].append(sum(crimes['value']))
```

In [11]:

```python
plt.figure(figsize=(13, 5))

xy = None

for year in crimes_per_month.keys():
    if year == '2008':
        xy = (0.1, max(crimes_per_month[year]))
        markevery = [0]
    elif year == '2012':
        xy = (2.1, max(crimes_per_month[year]))
        markevery = [2]
    else:
        xy = (6.1, max(crimes_per_month[year]))
        markevery = [6]

    plt.plot(range(len(crimes_per_month[year])), crimes_per_month[y
             label=year, alpha=.8, marker='o', markersize=7.0,
             markevery=markevery)
    plt.annotate(max(crimes_per_month[year]), xy)

plt.xticks(range(len(ds['month'].unique())),
           ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11'
plt.xlabel('Month')
plt.ylabel('No of Crimes')
plt.title('Most dangerous Years')
plt.grid()
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.tight_layout()
plt.savefig('../images/most_dangerous_years.pdf',
            bbox_inches='tight')
```



Summary of Most Dangerous Years

The figures above shows the flow of criminal activities on a by month basis for the most decrease years:

1. By looking at the criminal activities represented in this graphs like a flow, it is unique that the amount of criminal reports have the tendency to increase once every four years.

7. CATEGORICAL VARIABLE ANALYSIS

In [12]:
```
cropped_ds = ds.loc[ds['value'] != 0]
cropped_ds.describe(include=np.object)
```

Out[12]:

	lsoa_code	borough	major_category	minor_category	year	month
count	3419099	3419099	3419099	3419099	3419099	3419099
unique	4835	33	9	32	9	12
top	E01004734	Lambeth	Theft and Handling	Other Theft	2016	7
freq	2387	152784	1136994	297281	392042	296151

Summary of CATEGORICAL VARIABLE ANALYSIS

1. The year **2016** rise aside from numerical analysis.
2. Despite being the least decrease of criminal activities in the top three represented by the years, in descending order, **2008**, **2012** and **2016**, is the one that owns the majority of the records in the cropped dataset.
3. It means that, remaining coherent with what rise in the numeric variable's analysis, it has the lower crime per month ratio among the three.

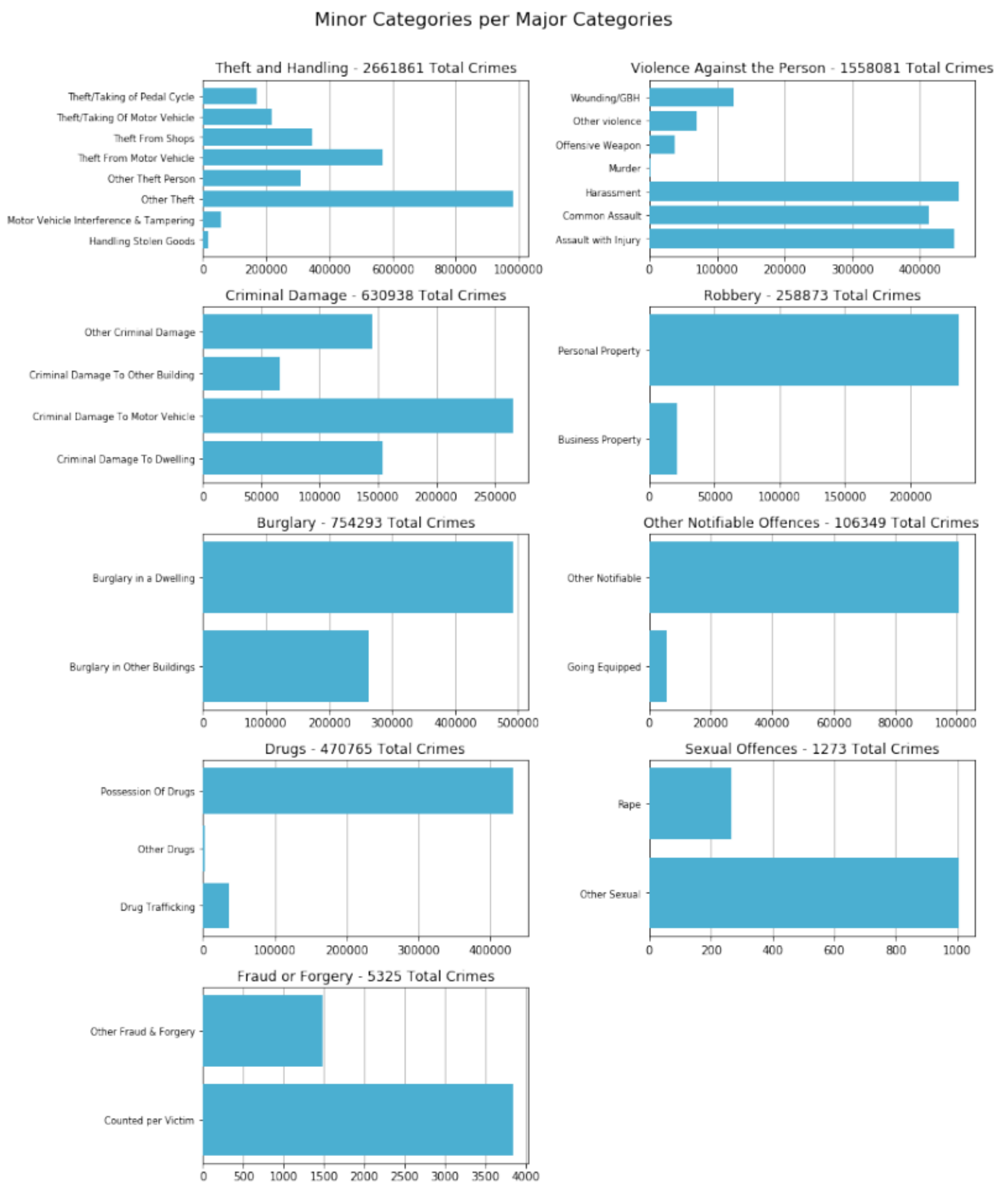
7.1 Visualizations For Minor Categories Per major_category

In [13]:
```
for major_category in cropped_ds['major_category'].unique():
    minor_categories = \
        list(cropped_ds.loc[cropped_ds['major_category'] == major_ca
        ['minor_category'].unique())
    print('\n{}: {}'.format(major_category, '\n\t'.join(minor_categ
```

```
Theft and Handling: Theft/Taking of Pedal Cycle
        Other Theft Person
        Other Theft
        Theft/Taking Of Motor Vehicle
        Theft From Shops
        Motor Vehicle Interference & Tampering
        Theft From Motor Vehicle
        Handling Stolen Goods

Violence Against the Person: Harassment
        Wounding/GBH
        Assault with Injury
        Common Assault
        Offensive Weapon
        Other violence
        Murder

Criminal Damage: Criminal Damage To Motor Vehicle
        Criminal Damage To Dwelling
        Criminal Damage To Other Building
        Other Criminal Damage

Robbery: Personal Property
        Business Property

Burglary: Burglary in a Dwelling
        Burglary in Other Buildings

Other Notifiable Offences: Going Equipped
        Other Notifiable

Drugs: Possession Of Drugs
        Drug Trafficking
        Other Drugs

Sexual Offences: Other Sexual
        Rape

Fraud or Forgery: Counted per Victim
        Other Fraud & Forgery
```

In [14]:
```
plt.figure(figsize=(12, 15))

for i, major_category in enumerate(cropped_ds['major_category'].uni
    min_cat = cropped_ds.loc[cropped_ds['major_category'] == major_
        groupby('minor_category').sum().to_dict()['value']

    plt.subplot(5, 2, i + 1)

    plt.barh(range(len(list(min_cat.keys()))), list(min_cat.values(
```

```
    plt.barh(range(len(list(min_cat.keys()))), list(min_cat.values(
             color='#4bafd1', align='center', zorder=3)
    plt.yticks(range(len(list(min_cat.keys()))), list(min_cat.keys(
               fontsize=8)
    plt.title('{} - {} Total Crimes'.format(major_category, sum(lis
              fontsize=12)
    plt.grid(zorder=0, axis='x')

plt.suptitle('Minor Categories per Major Categories', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.savefig('../images/minor_categories_per_major_categories.pdf',
            bbox_inches='tight')
```



Summary of Minor Categories Per major_category

1. The **minor category crimes** classification is very rich, with Theft and Handling being the most diversified with eight minor categories.
2. The subclass of the total number of criminal activities for each major category crime among its minor categories.
3. By observing the graphs it is possible to extract the most frequent minor category for each major category:
    - Theft and Handling -> Other Theft
    - Violence Against the Person -> Harrasment
    - Criminal Damage -> Criminal Damage To Motor Vehicle
    - Robbery -> Personal Robbery
    - Burglary -> Burglary in a Dwelling
    - Other Notifiable Offences -> Other Notifiable
    - Drugs -> Possession Of Drugs
    - Sexual Offences -> Other Sexual
    - Fraud or Forgery -> Counted per Victim

7.2 Visualizations for the three categorical variables borough,major_category and minor_category

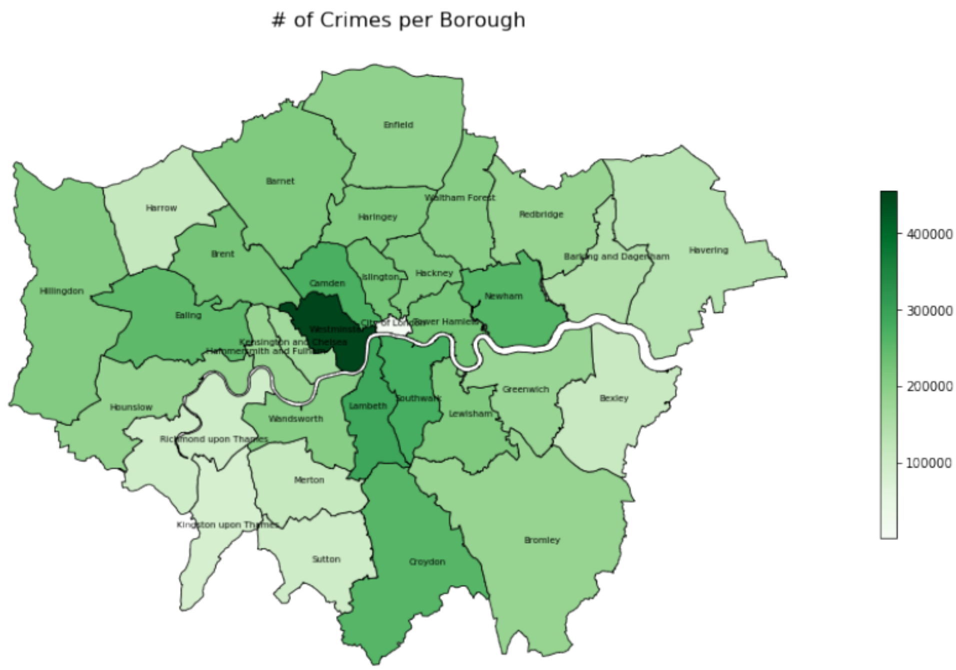
In [15]:
```
def plot_ordered_horizontal_barplot(dataset, column, title, x_label
                                    save=False):
    unique_values = dataset[column].unique()
    values_dict = defaultdict(list)

    for val in unique_values:
        values_dict[sum(dataset.loc[dataset[column] == val]['value'
            append(val)

    ys = range(sum([len(l) for l in values_dict.values()]))
    widths = []

    for key in values_dict:
        for value in values_dict[key]:
            widths.append(key)

    widths = np.sort(widths)

    ordered_keys = []

    for width in np.sort(list(values_dict.keys())):
        for value in values_dict[width]:
            ordered_keys.append(value)

    plt.grid(axis='x', zorder=0)
    plt.barh(y=ys, width=widths, align='center', zorder=3)
    plt.yticks(ys, ordered_keys)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(title)

    if save:
        plt.savefig(fname='../images/{}.pdf'.format(column),
                    bbox_inches='tight')
```

In [16]:
```
# London Borough/District Dataset - https://data.gov.uk/dataset/a76
pop_df = pd.read_csv('../data/housing-density-borough.csv')
```

In [17]:
```
# London Borough/District Map - https://www.kaggle.com/csobral/lond
map_df = gpd.read_file('../map/London_Borough_Excluding_MHW.shp')
```

In [18]:
```
crimes_per_borough = cropped_ds.groupby('borough')['value'].sum()
merged = map_df.set_index('NAME').join(crimes_per_borough)

merged['coords'] = merged['geometry'].\
    apply(lambda x: x.representative_point().coords[:])
merged['coords'] = [coords[0] for coords in merged['coords']]
```

```
In [19]: merged.plot(column='value', cmap='Greens', linewidth=0.8, edgecolor
                    figsize=(15, 10))
         plt.axis('off')
         plt.title('# of Crimes per Borough', fontsize=16)
         sm = plt.cm.ScalarMappable(cmap='Greens',
                                    norm=plt.Normalize(vmin=min(merged['valu
                                                       vmax=max(merged['valu
         sm._A = []
         cbar = plt.colorbar(sm, shrink=0.5)
         for idx, row in merged.iterrows():
             plt.annotate(s=idx, xy=row['coords'],
                          horizontalalignment='center', fontsize=7)
         plt.savefig(fname='../images/crimes_by_borough.pdf',
                     bbox_inches='tight')
```



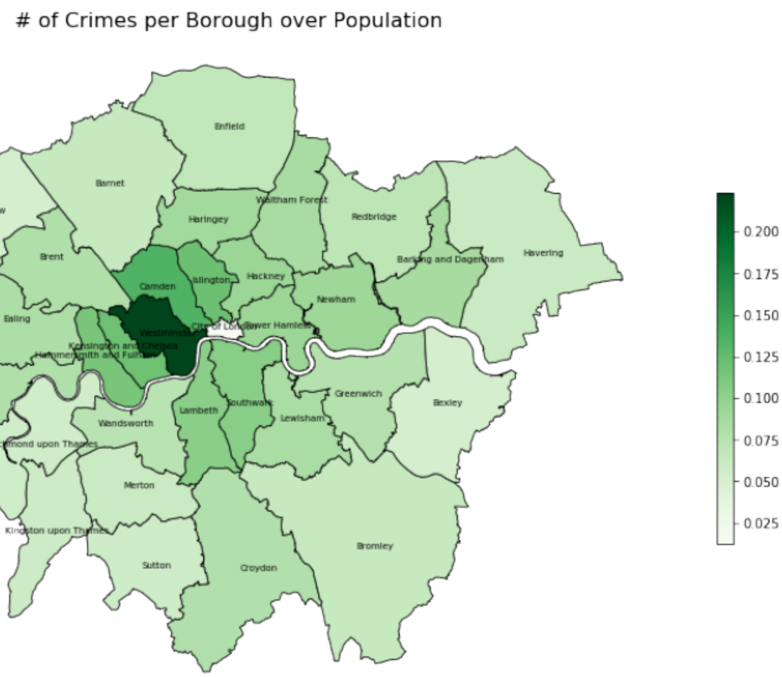
Summary of three categorical variables borough,major_category and minor_category

1. From the geographic visualization proves what has been discovered so far.
2. Westminster is confirmed as the most decrease of criminal activities among the boroughs, while City of London is confirmed as the least dense of criminal activities.
3. The naive assumption that there is no correlation between the number of crimes committed during the window of time proposed by the dataset and the **boroughs** territorial extension.

7.3 Visualizations for the categorical variables borough over Population

In [20]:
```
crimes_per_borough = crimes_per_borough.astype(float)

for borough in crimes_per_borough.index.tolist():
    crimes_per_borough[borough] = 0
    scores = []

    for year in range(2008, 2017):
        crimes = cropped_ds[(cropped_ds['borough'] == borough) &
                            (cropped_ds['year'] == str(year))]['valu
        population = pop_df[(pop_df['Name'] == borough) &
                            (pop_df['Year'] == year)]['Population']
        scores.append(crimes / population)

    crimes_per_borough[borough] = np.mean(scores)
```

In [21]:
```python
merged = map_df.set_index('NAME').join(crimes_per_borough)

merged['coords'] = merged['geometry'].\
    apply(lambda x: x.representative_point().coords[:])
merged['coords'] = [coords[0] for coords in merged['coords']]

merged.plot(column='value', cmap='Greens', linewidth=0.8, edgecolor
            figsize=(15, 10))
plt.axis('off')
plt.title('# of Crimes per Borough over Population', fontsize=16)
sm = plt.cm.ScalarMappable(cmap='Greens',
                           norm=plt.Normalize(vmin=min(merged['valu
                                              vmax=max(merged['valu
sm._A = []
cbar = plt.colorbar(sm, shrink=0.5)
for idx, row in merged.iterrows():
    plt.annotate(s=idx, xy=row['coords'],
                 horizontalalignment='center', fontsize=7)
plt.savefig(fname='../images/crimes_by_borough_over_pop.pdf',
            bbox_inches='tight')
```

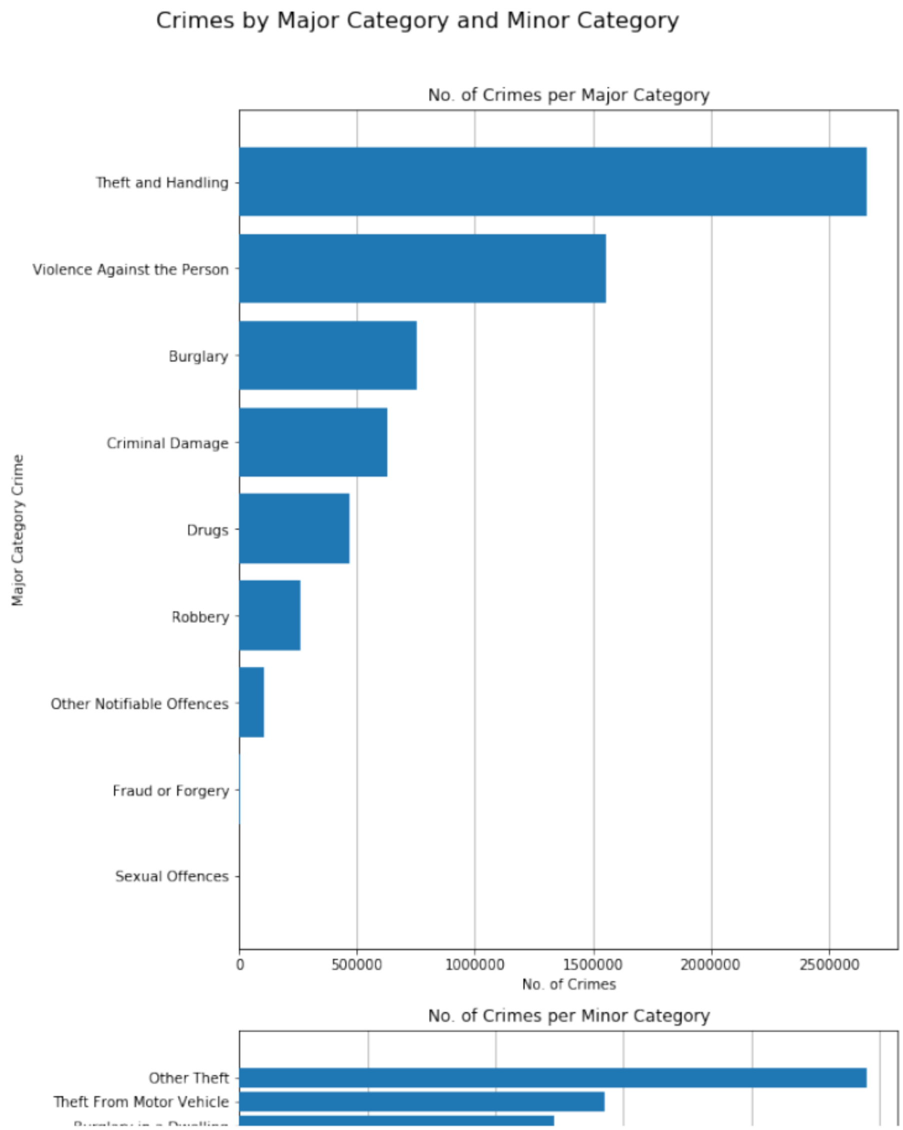


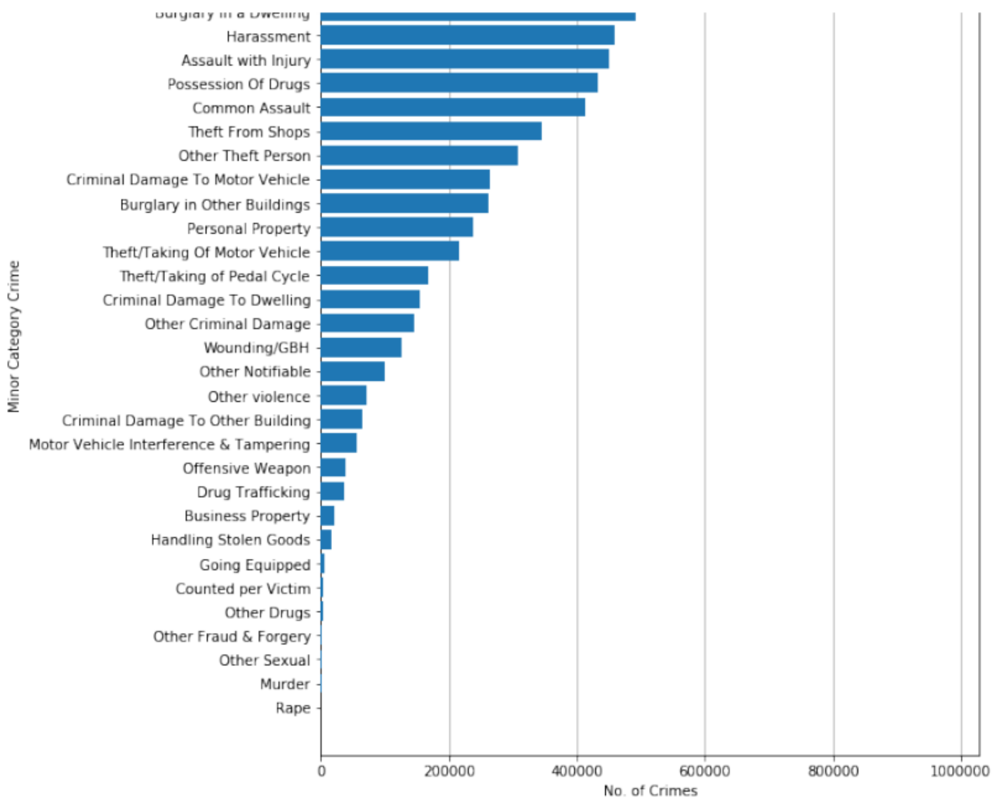
Summary of categorical variables borough over Population

1. This visualization shows a general very low score for the ratio between the number of crimes committed in a district and its population.
2. This means that, for a certain window of time, the number of criminal activities are fewer than the population density - in a way confirms the fact that the period of time investigated by the dataset is a quite safe window of time.

7.4 Visualizations for the categorical variables major_category and minor_category

In [22]:
```
plt.figure(figsize=(10, 20))

plt.subplot(2, 1, 1)
plot_ordered_horizontal_barplot(cropped_ds, 'major_category',
                                'No. of Crimes per Major Category',
                                'No. of Crimes', 'Major Category Cr
plt.subplot(2, 1, 2)
plot_ordered_horizontal_barplot(cropped_ds, 'minor_category',
                                'No. of Crimes per Minor Category',
                                'Minor Category Crime')
plt.suptitle('Crimes by Major Category and Minor Category',
             fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.savefig(fname='../images/crimes_by_borough_major_minor.pdf',
            bbox_inches='tight')
```





Summary of categorical variables major_category and minor_category

1. Despite being Lambeth the most popular borough among the cropped dataset's records, the most dangerous is actually Westminster, as depicted in the visualizations.
2. Theft and Handling is the most frequent major category crime and Other Theft is the most frequent minor category crime.

8. CORRELATION ANALYSIS

Since the majority of the dataset's variables are categorical variables, the **Pearson's chi-squared test** is been used.

In [23]:
```
def test_dependence(col_1, col_2, prob=0.95):
    contingency_table = pd.crosstab(cropped_ds[col_1],
                                    cropped_ds[col_2]).fillna(0)
    stat, p, dof, expected = stats.chi2_contingency(contingency_tab
    critical = stats.chi2.ppf(prob, dof)

    if abs(stat) >= critical:
        return 'Dependent'
    else:
        return 'Independent'
```

In [24]:
```python
correlation_data = np.empty((len(cropped_ds.columns), len(cropped_d:
columns = cropped_ds.columns.tolist()

for index, col_1 in enumerate(columns):
    c = index

    for col_2 in columns[index:]:
        result = test_dependence(col_1, col_2)
        correlation_data[index][c], correlation_data[c][index] = re:
        c += 1
```

In [25]:
```python
correlation_dataframe = pd.DataFrame(data=correlation_data, index=c(
correlation_dataframe
```

Out[25]:

	lsoa_code	borough	major_category	minor_category	value	
lsoa_code	Dependent	Dependent	Dependent	Dependent	Dependent	Depen
borough	Dependent	Dependent	Dependent	Dependent	Dependent	Depen
major_category	Dependent	Dependent	Dependent	Dependent	Dependent	Depen
minor_category	Dependent	Dependent	Dependent	Dependent	Dependent	Depen
value	Dependent	Dependent	Dependent	Dependent	Dependent	Depen
year	Dependent	Dependent	Dependent	Dependent	Dependent	Depen
month	Independent	Dependent	Dependent	Dependent	Dependent	Depen

Summary of CORRELATION ANALYSIS

1. The results returned by the correlation analysis are not surprising as expected.
2. The dataset is composed by a set of variables that are all **depending** on each other.
3. In the correlation table above, the majority of variables have a relation with the other variables that can be classified as **dependent**, while the variables **lsoa_code** and **month** are classified as **independent**.

8. CONCLUSION

1. Lambeth the most popular borough among the cropped dataset's records.
2. The most dangerous is actually Westminster, as depicted in the visualizations.
3. Theft and Handling is the most frequent major category crime and Other Theft is the most frequent minor category crime.
4. The variables in datasets are all depending on each other, the majority of variables have a relation with the other variables that can be classified as dependent, while the variables lsoa_code and month are classified as independent.

In [ ]:
```python
```

WQD7003_Group_Assingment

ORIGINALITY REPORT

22%	8%	7%	20%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to King's College Student Paper	2%
2	Submitted to Monash University Student Paper	1%
3	Submitted to University of Leeds Student Paper	1%
4	Submitted to University College London Student Paper	1%
5	Submitted to National College of Ireland Student Paper	1%
6	Submitted to Birkbeck College Student Paper	1%
7	www.coursehero.com Internet Source	1%
8	T. Ilavarasan, M. Meenakshi. "An overview of fiber dispersion and nonlinearity compensation techniques in optical orthogonal frequency division multiplexing systems", Journal of	1%

Optics, 2015
Publication

9	Submitted to Loughborough University Student Paper	1%
10	Submitted to Imperial College of Science, Technology and Medicine Student Paper	1%
11	Submitted to University of Southampton Student Paper	1%
12	datainquisitive.net Internet Source	1%
13	Submitted to Escuela Politecnica Nacional Student Paper	1%
14	Submitted to University of Sheffield Student Paper	1%
15	Submitted to City University Student Paper	1%
16	Submitted to University of Bristol Student Paper	1%
17	irukanobox.blogspot.com Internet Source	1%
18	Manohar Swamynathan. "Mastering Machine Learning with Python in Six Steps", Springer Science and Business Media LLC, 2019 Publication	<1%

19	democracy.towerhamlets.gov.uk Internet Source	<1%
20	www.downloads.lambethcpcg.org.uk Internet Source	<1%
21	Submitted to Infile Student Paper	<1%
22	Submitted to UC, Boulder Student Paper	<1%
23	ccforum.biomedcentral.com Internet Source	<1%
24	ikee.lib.auth.gr Internet Source	<1%
25	Submitted to University of Reading Student Paper	<1%
26	Submitted to University of Westminster Student Paper	<1%
27	Submitted to Hong Kong University of Science and Technology Student Paper	<1%
28	Submitted to Harrisburg University of Science and Technology Student Paper	<1%
29	www.co.randolph.nc.us Internet Source	<1%

30	Submitted to University of Nottingham Student Paper	<1%