London_Crime_Exam

September 29, 2021

1 London Crime

1.1 Introduction

This data counts the number of crimes at two different geographic levels of London (LSOA and borough) by year, according to crime type. Includes data from 2008 to 2016.
This public dataset is hosted in Google BigQuery and the table has the size of 1GB.

The **aim of the work** is to explore the data through the use of BigQuery and to predict the number of crimes for a specific year, given the information about the boroughs, the total number of codes for each borough and the different major categories of crime.

[1]:
```
#import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
```

[2]:
```
from google.cloud import bigquery

#Create a client project
PROJECT_ID= 'london-crime-exam-cc'
client = bigquery.Client(project=PROJECT_ID)
```

[3]:
```
dataset= """
SELECT *
FROM   bigquery-public-data.london_crime.crime_by_lsoa
"""
df = client.query(dataset).to_dataframe()
df.head()
```

[3]:
```
   lsoa_code    borough major_category               minor_category  value  \
0  E01002702  Islington       Burglary  Burglary in Other Buildings      1
1  E01002780  Islington       Burglary  Burglary in Other Buildings      0
2  E01002730  Islington       Burglary  Burglary in Other Buildings      0
3  E01033490  Islington       Burglary  Burglary in Other Buildings      1
4  E01002694  Islington       Burglary  Burglary in Other Buildings      1
```

```
   year  month
0  2016     12
1  2010     11
2  2015      8
3  2016      6
4  2011      8
```

The dataset is composed by 13M rows and 7 colums:

- *lsoa_code* (String): code for Lower Super Output Area in Greater London
- *borough* (String): Common name for London borough
- *major_category* (String): High level categorization of crime
- *minor_category* (String): Low level categorization of crime within major category
- *value* (Integer): Summary of the numbers of crimes for the month
- *year* (Integer): Year of reported counts, 2008-2016
- *month* (Integer): Month of reported counts, 1-12

[4]:
```python
df.info()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13490604 entries, 0 to 13490603
Data columns (total 7 columns):
 #   Column          Dtype
---  ------          -----
 0   lsoa_code       object
 1   borough         object
 2   major_category  object
 3   minor_category  object
 4   value           int64
 5   year            int64
 6   month           int64
dtypes: int64(3), object(4)
memory usage: 720.5+ MB
```

[4]:
```
lsoa_code         0
borough           0
major_category    0
minor_category    0
value             0
year              0
month             0
dtype: int64
```

In the dataset we do not have null values.

1.2 Exploratory Data Analysis

1.2.1 LSOA

Lower Layer Super Output Areas (LSOA) are a geographic hierarchy designed to improve the reporting of small area statistics in England and Wales.
They are built from groups of contiguous Output Areas and have been automatically generated to be as consistent in population size as possible, and typically contain from four to six Output Areas. *The Minimum population* is **1000** and the *mean* is **1500.**

[5]:
```
df['lsoa_code'].nunique()
```

[5]:
```
4835
```

How many Lsoa-code per borought? Thanks to this query we can understand the size of each borough which will be useful for interpreting subsequent results.
As we can see, the borough of **Croydon** turns out to be the one with the greatest number of codes unlike the City of London which has only 6 codes.

[6]:
```
lsoa_boroughs = """
SELECT Borough, count( DISTINCT lsoa_code) AS N_codes
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY borough
ORDER BY count(DISTINCT lsoa_code) desc;
        """

# Perform the query and store the result
lsoa_boroughs = client.query(lsoa_boroughs).to_dataframe()
lsoa_boroughs
```

[6]:

	Borough	N_codes
0	Croydon	220
1	Barnet	211
2	Bromley	197
3	Ealing	196
4	Enfield	183
5	Wandsworth	179
6	Lambeth	178
7	Brent	173
8	Lewisham	169
9	Southwark	166
10	Newham	164
11	Redbridge	161
12	Hillingdon	161
13	Greenwich	151
14	Havering	150
15	Bexley	146
16	Haringey	145

17	Hackney	144
18	Tower Hamlets	144
19	Waltham Forest	144
20	Hounslow	142
21	Harrow	137
22	Camden	133
23	Westminster	128
24	Merton	124
25	Islington	123
26	Sutton	121
27	Richmond upon Thames	115
28	Hammersmith and Fulham	113
29	Barking and Dagenham	110
30	Kensington and Chelsea	103
31	Kingston upon Thames	98
32	City of London	6

1.2.2 Borough

London boroughs are the administrative subdivisions of the neighborhood into which the British metropolis is divided; each of them has its own mayor and its own council.

[7]:
```
boroughs = """
SELECT DISTINCT Borough
FROM `bigquery-public-data.london_crime.crime_by_lsoa`;
        """

# Perform the query and store the result
boroughs = client.query(boroughs).to_dataframe()
boroughs
```

[7]:

	Borough
0	City of London
1	Brent
2	Barnet
3	Bexley
4	Camden
5	Ealing
6	Harrow
7	Merton
8	Newham
9	Sutton
10	Bromley
11	Croydon
12	Enfield
13	Hackney
14	Lambeth

```
15                 Haringey
16                 Havering
17                 Hounslow
18                 Lewisham
19                Greenwich
20                Islington
21                Redbridge
22                Southwark
23               Hillingdon
24               Wandsworth
25              Westminster
26            Tower Hamlets
27           Waltham Forest
28     Barking and Dagenham
29     Kingston upon Thames
30     Richmond upon Thames
31   Hammersmith and Fulham
32   Kensington and Chelsea
```

How many crimes there are per borough?

[8]:
```
crimes_per_borough = """
SELECT Borough, sum ( value) as Crimes
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY borough
ORDER BY sum( value) desc;
        """

crimes_per_borough = client.query(crimes_per_borough).to_dataframe()
crimes_per_borough
```

[8]:

	Borough	Crimes
0	Westminster	455028
1	Lambeth	292178
2	Southwark	278809
3	Camden	275147
4	Newham	262024
5	Croydon	260294
6	Ealing	251562
7	Islington	230286
8	Tower Hamlets	228613
9	Brent	227551
10	Hackney	217119
11	Lewisham	215137
12	Haringey	213272
13	Barnet	212191
14	Hillingdon	209680

15	Wandsworth	204741
16	Waltham Forest	203879
17	Enfield	193880
18	Hounslow	186772
19	Hammersmith and Fulham	185259
20	Bromley	184349
21	Redbridge	183562
22	Greenwich	181568
23	Kensington and Chelsea	171981
24	Barking and Dagenham	149447
25	Havering	138947
26	Harrow	116848
27	Merton	115654
28	Bexley	114136
29	Sutton	100987
30	Richmond upon Thames	96771
31	Kingston upon Thames	89306
32	City of London	780

Are the crimes more common in the most populous boroughs? Westminster , **Lambeth**, **Southwark**, **Camden** , **Newham** and **Croydon** result to be the Borough with the most committed crimes, but are these city also the most popular and so we could think to a relashionship between the population dimensions and the committed crimes.

[9]:
```
crime_code_per_borough = """
SELECT Borough, sum ( value) as Crimes, count( DISTINCT lsoa_code) AS N_codes
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY borough
ORDER BY N_codes desc;
        """

crime_code_per_borough = client.query(crime_code_per_borough).to_dataframe()
crime_code_per_borough
```

[9]:

	Borough	Crimes	N_codes
0	Croydon	260294	220
1	Barnet	212191	211
2	Bromley	184349	197
3	Ealing	251562	196
4	Enfield	193880	183
5	Wandsworth	204741	179
6	Lambeth	292178	178
7	Brent	227551	173
8	Lewisham	215137	169
9	Southwark	278809	166
10	Newham	262024	164
11	Redbridge	183562	161

```
12              Hillingdon  209680      161
13               Greenwich  181568      151
14                Havering  138947      150
15                  Bexley  114136      146
16                Haringey  213272      145
17                 Hackney  217119      144
18           Tower Hamlets  228613      144
19          Waltham Forest  203879      144
20                Hounslow  186772      142
21                  Harrow  116848      137
22                  Camden  275147      133
23             Westminster  455028      128
24                  Merton  115654      124
25               Islington  230286      123
26                  Sutton  100987      121
27    Richmond upon Thames   96771      115
28  Hammersmith and Fulham  185259      113
29    Barking and Dagenham  149447      110
30  Kensington and Chelsea  171981      103
31    Kingston upon Thames   89306       98
32          City of London     780        6
```

1.2.3 Major Category

[10]:
```
major_c = """
SELECT DISTINCT Major_Category
FROM `bigquery-public-data.london_crime.crime_by_lsoa`;
        """

# Perform the query and store the result
major_category = client.query(major_c).to_dataframe()
major_category
```

[10]:
```
              Major_Category
0  Violence Against the Person
1           Theft and Handling
2                        Drugs
3    Other Notifiable Offences
4                      Robbery
5              Criminal Damage
6                     Burglary
7              Sexual Offences
8            Fraud or Forgery
```

[11]:
```
major_c_count = """
SELECT Major_Category, sum ( value) as Crimes
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
```

```
GROUP BY Major_Category
ORDER BY Crimes desc;
        """

crimes_per_major_c = client.query(major_c_count).to_dataframe()
crimes_per_major_c
```

[11]:

```
                Major_Category   Crimes
0           Theft and Handling  2661861
1  Violence Against the Person  1558081
2                     Burglary   754293
3              Criminal Damage   630938
4                        Drugs   470765
5                      Robbery   258873
6    Other Notifiable Offences   106349
7             Fraud or Forgery     5325
8              Sexual Offences     1273
```

What are the top three Major Category for each Borough? Both this query and the one regarding the Minor Category, that will see later, can be usefull to the administration and management of police security plans to create or improve key departments, suitable for the most widespread crimes.

[12]:

```
most_common_crime = """
SELECT
  Borough,Major_category,Crimes
FROM (SELECT borough,major_category,
    RANK() OVER(PARTITION BY borough ORDER BY sum(value) DESC) AS␣
 ↪rank_per_borough,
    sum(value) AS Crimes
  FROM `bigquery-public-data.london_crime.crime_by_lsoa`
  GROUP BY borough, major_category )
WHERE rank_per_borough <= 3
ORDER BY Borough Desc;
"""

most_common_crime = client.query(most_common_crime).to_dataframe()
most_common_crime
```

[12]:

```
           Borough               Major_category  Crimes
0      Westminster           Theft and Handling  277617
1      Westminster  Violence Against the Person   71448
2      Westminster                        Drugs   34031
3       Wandsworth           Theft and Handling   92523
4       Wandsworth  Violence Against the Person   45865
..             …                            …       …
94          Barnet  Violence Against the Person   46565
```

```
95                 Barnet                       Burglary   36981
96   Barking and Dagenham           Theft and Handling   50999
97   Barking and Dagenham  Violence Against the Person   43091
98   Barking and Dagenham              Criminal Damage   18888

[99 rows x 3 columns]
```

1.2.4 Minor Category

[13]:
```python
minor_c = """
SELECT DISTINCT Minor_Category
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
ORDER BY Minor_Category;
        """

# Perform the query and store the result
minor_c = client.query(minor_c).to_dataframe()
minor_c
```

[13]:
```
                            Minor_Category
0                      Assault with Injury
1               Burglary in Other Buildings
2                    Burglary in a Dwelling
3                         Business Property
4                            Common Assault
5                        Counted per Victim
6               Criminal Damage To Dwelling
7          Criminal Damage To Motor Vehicle
8          Criminal Damage To Other Building
9                          Drug Trafficking
10                           Going Equipped
11                     Handling Stolen Goods
12                               Harassment
13   Motor Vehicle Interference & Tampering
14                                   Murder
15                          Offensive Weapon
16                    Other Criminal Damage
17                              Other Drugs
18                     Other Fraud & Forgery
19                         Other Notifiable
20                             Other Sexual
21                              Other Theft
22                       Other Theft Person
23                          Other violence
24                        Personal Property
25                       Possession Of Drugs
26                                     Rape
```

```
27                 Theft From Motor Vehicle
28                         Theft From Shops
29            Theft/Taking Of Motor Vehicle
30              Theft/Taking of Pedal Cycle
31                              Wounding/GBH
```

[14]:
```
minor_c_count = """
SELECT Minor_Category, sum ( value) as Crimes
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY Minor_Category
ORDER BY Crimes desc;
        """

crimes_per_minor_c = client.query(minor_c_count).to_dataframe()
crimes_per_minor_c
```

[14]:

	Minor_Category	Crimes
0	Other Theft	980085
1	Theft From Motor Vehicle	569956
2	Burglary in a Dwelling	491282
3	Harassment	458124
4	Assault with Injury	451001
5	Possession Of Drugs	431948
6	Common Assault	413690
7	Theft From Shops	345142
8	Other Theft Person	308842
9	Criminal Damage To Motor Vehicle	265463
10	Burglary in Other Buildings	263011
11	Personal Property	237578
12	Theft/Taking Of Motor Vehicle	216538
13	Theft/Taking of Pedal Cycle	168974
14	Criminal Damage To Dwelling	154116
15	Other Criminal Damage	145356
16	Wounding/GBH	125556
17	Other Notifiable	100819
18	Other violence	70778
19	Criminal Damage To Other Building	66003
20	Motor Vehicle Interference & Tampering	56224
21	Offensive Weapon	37983
22	Drug Trafficking	35819
23	Business Property	21295
24	Handling Stolen Goods	16100
25	Going Equipped	5530
26	Counted per Victim	3840
27	Other Drugs	2998
28	Other Fraud & Forgery	1485
29	Other Sexual	1005

```
30                                    Murder     949
31                                      Rape     268
```

What are the top three Minor Category for each Borough?

[15]:
```
most_common_minor_crime = """
SELECT
  Borough,Minor_Category,Crimes
FROM (SELECT borough,Minor_Category,
    RANK() OVER(PARTITION BY borough ORDER BY sum(value) DESC) AS␣
 ↪rank_per_borough,
    sum(value) AS Crimes
  FROM `bigquery-public-data.london_crime.crime_by_lsoa`
  GROUP BY borough, Minor_Category )
WHERE rank_per_borough <= 3
ORDER BY Borough;
"""

most_common_minor_crime = client.query(most_common_minor_crime).to_dataframe()
most_common_minor_crime
```

[15]:
```
                 Borough            Minor_Category  Crimes
0   Barking and Dagenham               Other Theft   16740
1   Barking and Dagenham        Assault with Injury   13719
2   Barking and Dagenham     Burglary in a Dwelling   12885
3                 Barnet               Other Theft   29966
4                 Barnet     Burglary in a Dwelling   26165
..                   …                         …       …
94            Wandsworth  Theft From Motor Vehicle   22222
95            Wandsworth     Burglary in a Dwelling   15166
96           Westminster               Other Theft  142032
97           Westminster         Other Theft Person   56756
98           Westminster          Theft From Shops   35929

[99 rows x 3 columns]
```

1.2.5 Value, Year and Month

[16]:
```
df['year'].min(),df['year'].max()
```

[16]:
```
(2008, 2016)
```

[17]:
```
value_c = """
SELECT  Year, Month, sum(value) as Crimes
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY year, month
ORDER BY year,month asc;
```

```
"""

# Perform the query and store the result
value_c = client.query(value_c).to_dataframe()
value_c
```

[17]:

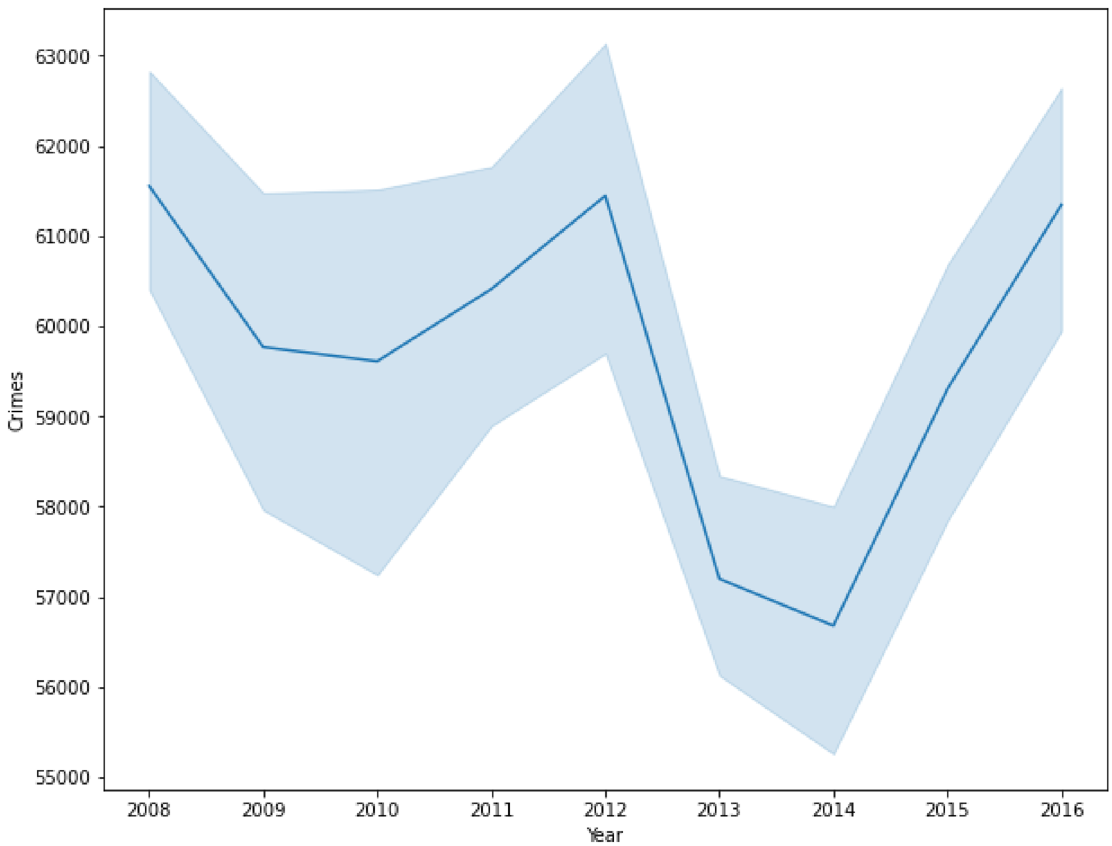
```
     Year  Month  Crimes
0    2008      1   65419
1    2008      2   62626
2    2008      3   61343
3    2008      4   59640
4    2008      5   62587
..    …      …      …
103  2016      8   62666
104  2016      9   61412
105  2016     10   63405
106  2016     11   61064
107  2016     12   62455

[108 rows x 3 columns]
```

Thanks to the lineplot we can verify the trend of the occurrence of crimes along the time axis available to us. As we can see, the trend undergoes a sudden and abrupt decrease during the year 2013-2014. This could be due to an effective reduction in crime or to a failure to record some data, also due to the fact that from 2014 onwards we are witnessing again an increase in crime which returns to around 61,000 cases.

[19]:

```
plt.figure(figsize=(10,8))
sns.lineplot(x='Year', y='Crimes', data= value_c)
plt.show()
```



How Growth Rate has change over the years for each Major Category?

[20]:
```
growing_crime = """
SELECT
    Year,
    Major_category,
    SUM(value) as `Crimes`,
    CASE WHEN SUM(value) > 0
        THEN SUM(value) / LAG(SUM(value) , 1 , NULL) OVER (PARTITION BY␣
 ↪major_category ORDER BY year ASC ) - 1
        ELSE 0 END AS `Growth_rate`
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY major_category, year
ORDER BY major_category, year DESC;
        """

growing_crime = client.query(growing_crime).to_dataframe()
growing_crime
```

[20]:
```
    Year               Major_category  Crimes  Growth_rate
0   2016                     Burglary   68285    -0.031267
1   2015                     Burglary   70489    -0.073160
2   2014                     Burglary   76053    -0.128053
3   2013                     Burglary   87222    -0.066066
4   2012                     Burglary   93392     0.000825
..   …                            …       …          …
76  2012  Violence Against the Person  150014     0.021191
77  2011  Violence Against the Person  146901    -0.069623
78  2010  Violence Against the Person  157894    -0.017932
79  2009  Violence Against the Person  160777     0.005837
80  2008  Violence Against the Person  159844          NaN

[81 rows x 4 columns]
```

1.3 Prediction

After having explore the data, I create a final table from which we can perform the *prediction* of the "Total Crime".

1.3.1 Final Table

The final dataset is composed by 5 variables: *Year*, *Borough*, *N_codes* (count for each borough), *Major_Category* and *Total_Crime*.
It is not necessary to perform a transformation of the categorical variable into numerical ones because BigQuery ML will automatically performs a One-Hot Encoding Transformation for all models,except the Boosted Tree models for which will be applied a **Label Encoding Transformation**.
This transformation convert each unique value into a numerical value.

[22]:
```
final_table = """
SELECT Year,Borough,count( DISTINCT lsoa_code) AS N_codes,Major_category,␣
 ↪sum(value) AS Total_Crime
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY Borough, year, major_category
ORDER BY year
"""
final_table = client.query(final_table).to_dataframe()
final_table
```

[22]:
```
      Year         Borough  N_codes               Major_category  Total_Crime
0     2008  City of London        6            Theft and Handling            0
1     2008  City of London        5               Criminal Damage            0
2     2008  City of London        5                      Burglary            0
3     2008           Brent      173  Violence Against the Person         5690
4     2008           Brent      173                         Drugs         2813
…      …              …        …                            …           …
2650  2016  City of London        6            Theft and Handling          129
```

```
2651  2016  City of London        4    Other Notifiable Offences            6
2652  2016  City of London        5              Criminal Damage            2
2653  2016  City of London        4                      Robbery            4
2654  2016  City of London        5                        Drugs           10

[2655 rows x 5 columns]
```

Gradient Boosting Regression *The Gradient Boosting Machine* is a powerful ensemble machine learning algorithm that uses decision trees.
Boosting is a general ensemble technique that involves sequentially adding models to the ensemble where subsequent models correct the performance of prior models.
"Gradient" because it uses a gradient descent algorithm to minimize the loss when adding new models.

The parameter chosen for the model are: - *Tree_Method* : Type of tree construction algorithm. HIST is recommended for large datasets in order to achieve faster training speed and lower resource consumption. - *Data_Split_Method_* : The method to split input data into training and evaluation sets. Training data is used to train the model. Evaluation data is used to avoid overfitting due to early stopping. **'AUTO_SPLIT'** The automatic split strategy is as follows: When there are fewer than 500 rows in the input data, all rows are used as training data. When there are between 500 and 50,000 rows in the input data, 20% of the data is used as evaluation data in a RANDOM split. When there are more than 50,000 rows in the input data, only 10,000 rows are used as evaluation data in a RANDOM split. - *Input_Label_Columns* : The label column name(s) in the training data.

[23]:
```
model_create = """
  CREATE OR REPLACE MODEL `bqml_model.Crime`
  OPTIONS(MODEL_TYPE='BOOSTED_TREE_REGRESSOR',
          TREE_METHOD = 'HIST',
         DATA_SPLIT_METHOD = 'AUTO_SPLIT',
        INPUT_LABEL_COLS = ['Total_Crime'])
  AS
SELECT Year,Borough,count( DISTINCT lsoa_code) AS N_codes,Major_category,␣
 ↪sum(value) AS Total_Crime
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY Borough, year, major_category
ORDER BY year
  """
model_create = client.query(model_create).to_dataframe()
model_create
```

[23]:
```
Empty DataFrame
Columns: []
Index: []
```

[24]:
```
model_evaluate = """
SELECT *
```

```
FROM  ML.EVALUATE(MODEL `bqml_model.Crime`,
    ( SELECT Year,Borough,count( DISTINCT lsoa_code) AS N_codes,Major_category,␣
 ↪sum(value) AS Total_Crime
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY Borough, year, major_category
ORDER BY year ))
  """
model_evaluate = client.query(model_evaluate).to_dataframe()
model_evaluate
```

[24]:
```
   mean_absolute_error  mean_squared_error  mean_squared_log_error  \
0           187.567068       106258.396159                0.805978

   median_absolute_error  r2_score  explained_variance
0              91.726562  0.990694            0.990696
```

- **Mean Absolute Error (MAE)** : is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. It is negatively-oriented scores, which means lower values are better. -**Mean Squared Error (MSE)** : is the average squared difference between the estimated values and the actual value. Value lies between 0 to infinite and small value indicates better model.Sensitive to outliers, punishes larger error more. -**Mean Squared Logarithmic Error (MSLE)**: it only cares about the percentual difference between the true and the predicted value. This means that MSLE will treat small differences between small true and predicted values approximately the same as big differences between large true and predicted values. -**Median Absolute Error** : is the median of all of the absolute values of the residuals, and for this it is essentially insensitive to outliers.
- **R Squared**: measures how much of variability in dependent variable can be explained by the model. Value are between 0 to 1 and bigger value indicates a better fit between prediction and actual value.
- **Explained Variance**: it's the part of the model's total variance that is explained by factors that are actually present and isn't due to error variance.

[28]:
```
model_predict = """
SELECT *
FROM ML.PREDICT(MODEL `bqml_model.Crime`,
    (SELECT Year,Borough,count( DISTINCT lsoa_code) AS N_codes,Major_category,␣
 ↪sum(value) AS Total_Crime
FROM `bigquery-public-data.london_crime.crime_by_lsoa`
GROUP BY Borough, year, major_category
ORDER BY year))
ORDER BY year DESC
  """
model_predict = client.query(model_predict).to_dataframe()
model_predict
```

[28]:
```
      predicted_Total_Crime  Year                 Borough  N_codes  \
0                261.638397  2016          City of London        6
1                227.378769  2016          City of London        6
2                 30.957668  2016          City of London        5
3                  6.991164  2016          City of London        4
4                  6.991164  2016          City of London        4
…                       …     …                       …      …
2650             282.161591  2008  Kensington and Chelsea      103
2651             515.707397  2008  Kensington and Chelsea      103
2652             144.156693  2008  Kensington and Chelsea       46
2653            1801.948608  2008  Kensington and Chelsea      103
2654            1881.435303  2008  Kensington and Chelsea      103

                   Major_category  Total_Crime
0     Violence Against the Person           25
1              Theft and Handling          129
2                           Drugs           10
3       Other Notifiable Offences            6
4                         Robbery            4
…                             …          …
2650    Other Notifiable Offences          253
2651                      Robbery          530
2652              Fraud or Forgery           85
2653              Criminal Damage         1413
2654                     Burglary         1783

[2655 rows x 6 columns]
```