HarvardX Capstone: London Crime Project

Somosree Banerjee

21/06/2020

I. Introduction

This report is part of the final capstone project of the EdX course "HarvardX: PH125.9x Data Science: Capstone". The goal is to challenge and demonstrate how the knowledge acquired through the different topics covered in "HarvardX: PH125.9x Data Science" can be applied in solving real world problems. For this project, the London Crime data spanning 2008-2016 has been considered from the source:https://www.kaggle.com/jboysen/london-crime. The entire report will step by step explain the approach of data analysis and machine algorithm on the London Crime data set.

II. Summary

For the London Crime project, the data set provided has been taken from Kaggle.The aim is to create a recommendation system using the "prediction version of problem". The report has been split in three sections: 1. Data Loading 2. Data Visualization & Exploration 3. Machine Learning Algorithm for predicting a model

III. Data Loading

Memory has been set and the garbage collection has been excuted with respect to the current active session,

```
#Memory
memory.limit()
```

```
## [1] 8031
```

```
memory.limit(size=560000)
```

```
## [1] 560000
```

```
gc()
```

```
##           used (Mb) gc trigger (Mb) max used (Mb)
## Ncells  544948 29.2    1246328 66.6   621331 33.2
## Vcells 1031861  7.9    8388608 64.0  1600464 12.3
```

```
rm()
```

This section consist of the data loading details and the creation of training and test data corresponding to the London Crime data set. It should be noted that the entire analysis,visualization and prediction model determination using machine learning has been done on the training set. The validation of the prediction model has been carried on with the corresponding to the validation set.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## <U+2713> ggplot2 3.3.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.6.3

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
if(!require(splitstackshape)) install.packages("splitstackshape")
```

```
## Loading required package: splitstackshape
```

```
## Warning: package 'splitstackshape' was built under R version 3.6.3
```

```r
if(!require(DT)) install.packages("DT")
```

```
## Loading required package: DT
```

```
## Warning: package 'DT' was built under R version 3.6.3
```

```r
if(!require(lubridate)) install.packages("lubridate")
```

```
## Loading required package: lubridate
```

```
## Warning: package 'lubridate' was built under R version 3.6.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
if(!require(ggpubr)) install.packages("ggpubr")
```

```
## Loading required package: ggpubr
```

```
## Warning: package 'ggpubr' was built under R version 3.6.3
```

```r
if(!require(patchwork)) install.packages("patchwork")
```

```
## Loading required package: patchwork
```

```
## Warning: package 'patchwork' was built under R version 3.6.3
```

```r
if(!require(hrbrthemes)) install.packages("hrbrthemes")
```

```
## Loading required package: hrbrthemes
```

```
## Warning: package 'hrbrthemes' was built under R version 3.6.3
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and

##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```r
if(!require(scales)) install.packages("scales")
```

```
## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
if(!require(tidytext)) install.packages("tidytext")
```

```
## Loading required package: tidytext

## Warning: package 'tidytext' was built under R version 3.6.3
```

```r
if(!require(ggalt))install.packages("ggalt")
```

```
## Loading required package: ggalt

## Warning: package 'ggalt' was built under R version 3.6.3

## Registered S3 methods overwritten by 'ggalt':
##   method                  from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob  ggplot2
##   grobX.absoluteGrob      ggplot2
##   grobY.absoluteGrob      ggplot2
```

```r
if(!require(purrr))install.packages("purrr")
if(!require(randomForest))install.packages("randomForest")
```

```
## Loading required package: randomForest

## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Libraries
library(tidyverse)
library(caret)
library(data.table)
library(splitstackshape)
library(DT)
library(lubridate)
library(ggpubr)     ## Extra visualizations and themes
library(patchwork) ## Patch visualizations together
library(hrbrthemes)## extra themes and formatting
library(scales)     ## For formatting numeric variables
library(tidytext)  ## Reordering within facets in ggplot2
library(ggalt)      ## Extra visualizations
library(purrr)
library(randomForest)
```

The data has been loaded from Kaggle : https://www.kaggle.com/jboysen/london-crime and an additional field "Display_Date" (considering 1st Day of the Month) has been introduced in order to have a better visualization with the methods avilable in the available loaded libraries.

```
london_crimes <- read_csv("C:\\Somosree_BackUp\\Somosree\\DataScience\\Harvard\\Capstone\\LondonCrimePr
```

```
## Parsed with column specification:
## cols(
##   lsoa_code = col_character(),
##   borough = col_character(),
##   major_category = col_character(),
##   minor_category = col_character(),
##   value = col_double(),
##   year = col_double(),
##   month = col_double()
## )
```

```
london_crimes <- london_crimes %>% mutate(Display_Date = as.Date(paste(london_crimes$year, london_crime

## Show the first 6 rows
head(london_crimes)
```

```
## # A tibble: 6 x 8
##   lsoa_code borough major_category minor_category value  year month Display_Date
##   <chr>     <chr>   <chr>          <chr>          <dbl> <dbl> <dbl> <date>
## 1 E01001116 Croydon Burglary       Burglary in O...     0  2016    11 2016-11-01
## 2 E01001646 Greenw... Violence Agai... Other violence     0  2016    11 2016-11-01
## 3 E01000677 Bromley Violence Agai... Other violence     0  2015     5 2015-05-01
## 4 E01003774 Redbri... Burglary       Burglary in O...     0  2016     3 2016-03-01
## 5 E01004563 Wandsw... Robbery        Personal Prop...     0  2008     6 2008-06-01
## 6 E01001320 Ealing  Theft and Han... Other Theft        0  2012     5 2012-05-01
```

```
# Validation set will be 10% of data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = london_crimes$value, times = 1, p = 0.1, list = FALSE)
edx_london_crimes <- london_crimes[-test_index,]
temp_london_crimes <- london_crimes[test_index,]

# Make sure lsoa_code is avaible in main edx_london_crimes
validation <- temp_london_crimes %>%
  semi_join(edx_london_crimes, by = "lsoa_code")

# Add rows removed from validation set back into edx_london_crimes set
removed <- anti_join(temp_london_crimes, validation)
```

```
## Joining, by = c("lsoa_code", "borough", "major_category", "minor_category", "value", "year", "month"
```

```
edx_london_crimes <- rbind(edx_london_crimes, removed)

head(edx_london_crimes)
```

```
## # A tibble: 6 x 8
##   lsoa_code borough major_category minor_category value  year month Display_Date
##   <chr>     <chr>   <chr>          <chr>          <dbl> <dbl> <dbl> <date>
## 1 E01001116 Croydon Burglary       Burglary in O...     0  2016    11 2016-11-01
## 2 E01000677 Bromley Violence Agai... Other violence     0  2015     5 2015-05-01
## 3 E01003774 Redbri... Burglary       Burglary in O...     0  2016     3 2016-03-01
## 4 E01004563 Wandsw... Robbery        Personal Prop...     0  2008     6 2008-06-01
## 5 E01001320 Ealing  Theft and Han... Other Theft        0  2012     5 2012-05-01
## 6 E01002633 Hounsl... Robbery        Personal Prop...     0  2013     4 2013-04-01
```

III. Data Visualization & Exploration

In this section, various visualization methods have been implemented in order to analyze and explore the data so that a pattern of London Crime Count can be determined based on the available data set. Most of the data analysis representation has been potrayed using a tabular as well as graphical view.

A quantitative analysis has been conducted to undertand the yearly crime count across 2008-2016 and the corresponding percentage variation per year.

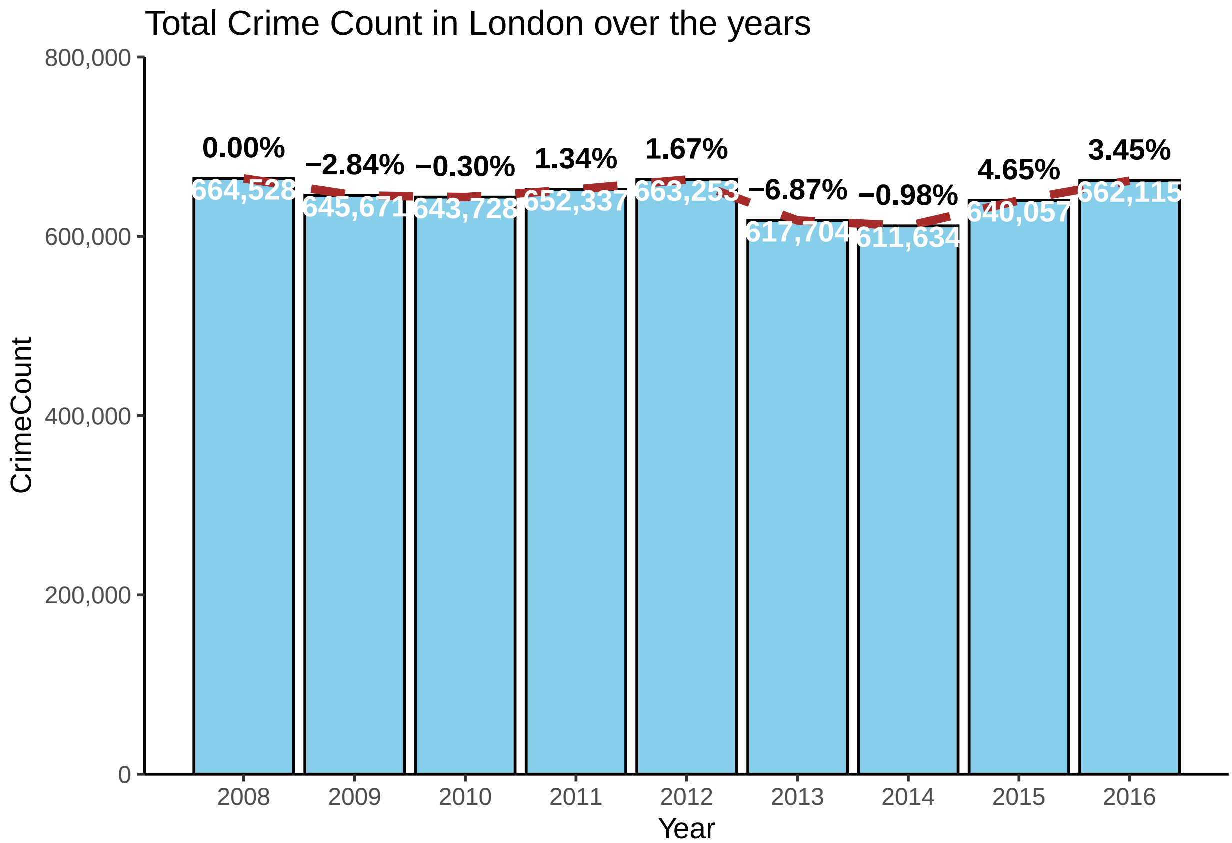
```r
#Tabular Representation
edx_Yearly_Crime_Count <- edx_london_crimes %>%
  group_by(Year = year) %>%
  summarise(CrimeCount=sum(value,na.rm = TRUE)) %>%
  ungroup() %>% mutate(Percent_Crime_Variation = (CrimeCount-lag(CrimeCount))/lag(CrimeCount),
                       Percent_Crime_Variation=replace_na(Percent_Crime_Variation,0)) %>%
  arrange(desc(CrimeCount))

datatable(edx_Yearly_Crime_Count, rownames = FALSE, filter="top", options = list(pageLength = 50, scrol
  formatRound('CrimeCount',digits=0, interval = 3, mark = ",") %>%
  formatRound('Percent_Crime_Variation',digits=3, interval = 3, mark = ",")
```

Show 50 entries Search:

Year	CrimeCount	Percent_Crime_Variation
All	All	All
2008	664,528	0.000
2012	663,253	0.017
2016	662,115	0.034
2011	652,337	0.013
2009	645,671	-0.028
2010	643,728	-0.003
2015	640,057	0.046
2013	617,704	-0.069
2014	611,634	-0.010

Showing 1 to 9 of 9 entries Previous 1 Next

```r
#Graphical Representation (Bar Graph)
edx_Yearly_Crime_Count <- edx_london_crimes %>%
  group_by(Year = floor_date(Display_Date,unit = "year")) %>%
  summarise(CrimeCount=sum(value,na.rm = TRUE)) %>%
  ungroup() %>% mutate(Percent_Crime_Variation = (CrimeCount-lag(CrimeCount))/lag(CrimeCount),
                       Percent_Crime_Variation=replace_na(Percent_Crime_Variation,0))

edx_Yearly_Crime_Count%>%
  ggplot(aes(Year,CrimeCount))+
  geom_bar(stat="identity",fill="skyblue",color="black")+
  geom_line(color="brown",size=1.5,linetype="dashed")+
  geom_text(aes(label=percent(Percent_Crime_Variation)),vjust=-1,color="black",fontface="bold")+  geom_t
  scale_y_comma(expand = c(0,0),limits = c(0,800000))+
  scale_x_date(breaks = "year",date_labels ="%Y")+
  theme_classic()+
  labs(title = "Total Crime Count in London over the years")
```



As per the analysis, 2014 has the least crime count and 2016 has the highest crime count.

It is require to understand how the crime count has been distributed across the different Boroughs and which has the highest crime count in the span 0f 2008-2016

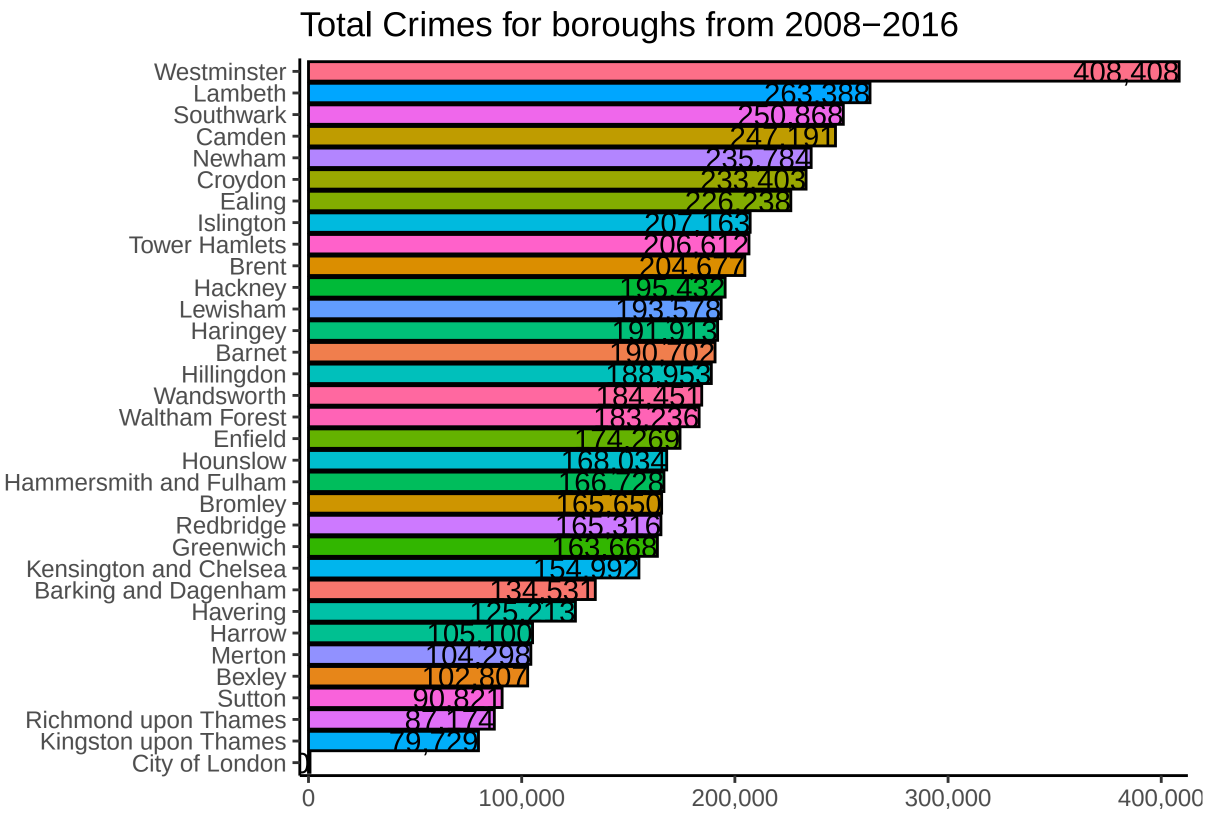
```
edx_london_crimes_borough <- edx_london_crimes %>%
  group_by(borough) %>%
  summarise(CrimeCount=sum(value))%>%
  arrange(desc(CrimeCount))%>%
  ungroup()
#Tabular Representation
datatable(edx_london_crimes_borough, rownames = FALSE, filter="top", options = list(pageLength = 50, sc
  formatRound('CrimeCount',digits=0, interval = 3, mark = ",")
```

Show 50 entries Search:

borough	CrimeCount
All	All
Westminster	408,408
Lambeth	263,388
Southwark	250,868
Camden	247,191
Newham	235,784
Croydon	233,403
Ealing	226,238
Islington	207,163
Tower Hamlets	206,612
Brent	204,677
Hackney	195,432
Lewisham	193,578
Haringey	191,913
Barnet	190,702
Hillingdon	188,953
Wandsworth	184,451
Waltham Forest	183,236
Enfield	174,269
Hounslow	168,034
Hammersmith and Fulham	166,728
Bromley	165,650
Redbridge	165,316
Greenwich	163,668
Kensington and Chelsea	154,992
Barking and Dagenham	134,531
Havering	125,213
Harrow	105,100
Merton	104,298
Bexley	102,807
Sutton	90,821
Richmond upon Thames	87,174
Kingston upon Thames	79,729
City of London	700

Showing 1 to 33 of 33 entries Previous 1 Next

```r
#Graphical Representation (Bar Graph)
edx_london_crimes_borough %>%
  ggplot(aes(reorder(borough,CrimeCount),CrimeCount))+
  geom_bar(stat = "identity",aes(fill=borough),color="black")+
  coord_flip()+
  scale_y_comma()+
  geom_text(aes(label=comma(CrimeCount)),hjust=1)+
  theme_classic()+
  theme(legend.position = "none")+
  labs(x=" ",y=" ",title = "Total Crimes for boroughs from 2008-2016 ")
```



As per the above visual representation, Westminster has the highest crime count in the span of 2008-2016.

Now, it is require to determine how the crime rate has increased and which borough has the highest increase in the years spanning of 2008 -2016.

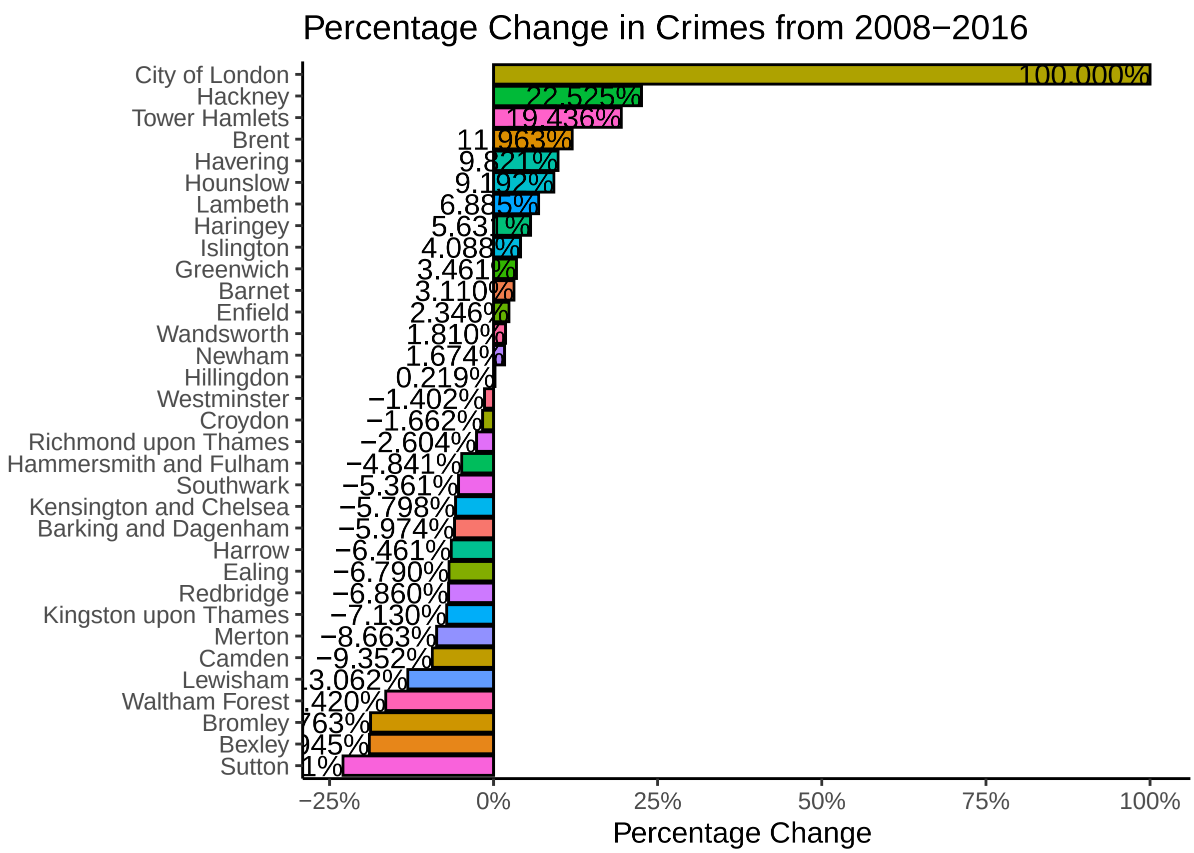
```r
edx_london_crimes_evolution <-edx_london_crimes %>%
  group_by(borough) %>% summarise(Crimes_2008=sum(value[year(Display_Date)==2008]),Crimes_2016=sum(valu
  ungroup() %>%
  mutate(CrimeRateIncPct=(Crimes_2016-Crimes_2008)/Crimes_2016) %>%
  arrange(desc(CrimeRateIncPct))

#Tabular Representation
datatable(edx_london_crimes_evolution, rownames = FALSE, filter="top", options = list(pageLength = 50, s
  formatRound('CrimeRateIncPct',digits=3, interval = 3, mark = ",")
```

Show 50 entries   Search:

borough	Crimes_2008	Crimes_2016	CrimeRateIncPct
All	All	All	All
City of London	0	158	1.000
Hackney	19784	25536	0.225
Tower Hamlets	21351	26502	0.194
Brent	21216	24099	0.120
Havering	13783	15284	0.098
Hounslow	18178	20018	0.092
Lambeth	28440	30543	0.069
Haringey	23126	24506	0.056
Islington	23389	24386	0.041
Greenwich	19832	20543	0.035
Barnet	21496	22186	0.031
Enfield	19773	20248	0.023
Wandsworth	20993	21380	0.018
Newham	26673	27127	0.017
Hillingdon	21847	21895	0.002
Westminster	43756	43151	-0.014
Croydon	26295	25865	-0.017
Richmond upon Thames	10047	9792	-0.026
Hammersmith and Fulham	19210	18323	-0.048
Southwark	30072	28542	-0.054
Kensington and Chelsea	18575	17557	-0.058
Barking and Dagenham	16002	15100	-0.060
Harrow	12522	11762	-0.065
Ealing	26029	24374	-0.068
Redbridge	18803	17596	-0.069
Kingston upon Thames	9511	8878	-0.071
Merton	12480	11485	-0.087
Camden	28379	25952	-0.094
Lewisham	24375	21559	-0.131
Waltham Forest	21604	18557	-0.164
Bromley	21489	18094	-0.188
Bexley	13794	11597	-0.189
Sutton	11704	9520	-0.229

Showing 1 to 33 of 33 entries   Previous 1 Next

```r
#Graphical Representation (Bar Graph)
edx_london_crimes_evolution%>%
  ggplot(aes(reorder(borough,CrimeRateIncPct),CrimeRateIncPct))+
  geom_bar(stat = "identity",aes(fill=borough),color="black")+
  coord_flip()+
  scale_y_continuous(labels = percent_format())+
  geom_text(aes(label=percent(CrimeRateIncPct)),hjust=1)+
  theme_classic()+
  theme(legend.position = "none")+
  labs(x=" ",y="Percentage Change ",title = "Percentage Change in Crimes from 2008-2016")
```



The graphical and tabular representation shows that London has highest crime increase (%) whereas WestMinster doesn't show any steep increase in crime count over the year 2008 -2016.

Next, the available data set has been visualized for Boroughs having the highest crime count and the corresponding year.

```r
edx_Max_Crimes_Borough_Year <- edx_london_crimes %>%
  group_by(borough,Year = year) %>%
  summarise(CrimeCount=sum(value)) %>%
  ungroup() %>%
  group_by(borough) %>%
  filter(CrimeCount==max(CrimeCount)) %>%
  ungroup() %>%
  arrange(desc(CrimeCount))
```

```
#Tabular Representation
datatable(edx_Max_Crimes_Borough_Year, rownames = FALSE, filter="top", options = list(pageLength = 50,
```

Show 50 entries

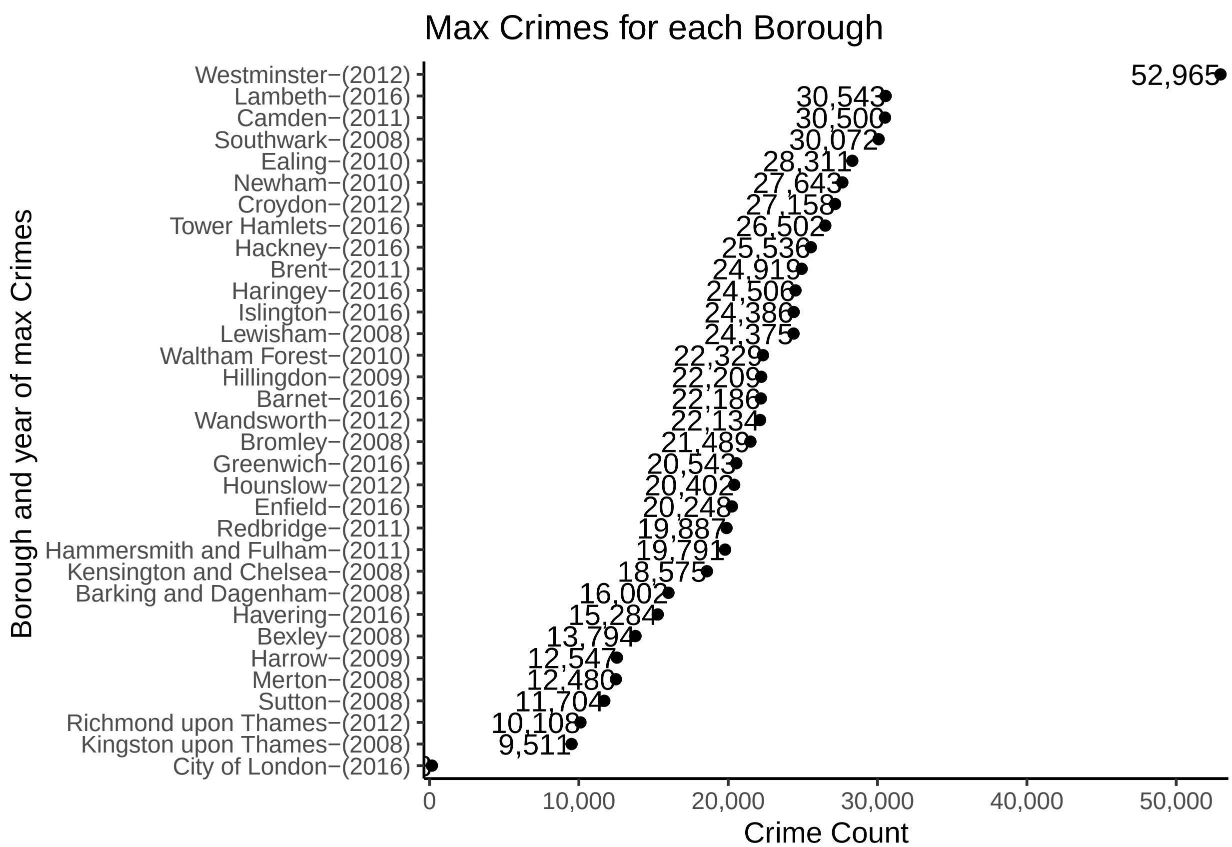
Search:

borough	Year	CrimeCount
All	All	All
Westminster	2012	52965
Lambeth	2016	30543
Camden	2011	30500
Southwark	2008	30072
Ealing	2010	28311
Newham	2010	27643
Croydon	2012	27158
Tower Hamlets	2016	26502
Hackney	2016	25536
Brent	2011	24919
Haringey	2016	24506
Islington	2016	24386
Lewisham	2008	24375
Waltham Forest	2010	22329
Hillingdon	2009	22209
Barnet	2016	22186
Wandsworth	2012	22134
Bromley	2008	21489
Greenwich	2016	20543
Hounslow	2012	20402
Enfield	2016	20248
Redbridge	2011	19887
Hammersmith and Fulham	2011	19791
Kensington and Chelsea	2008	18575
Barking and Dagenham	2008	16002
Havering	2016	15284
Bexley	2008	13794
Harrow	2009	12547
Merton	2008	12480
Sutton	2008	11704
Richmond upon Thames	2012	10108
Kingston upon Thames	2008	9511
City of London	2016	158

Showing 1 to 33 of 33 entries

Previous 1 Next

```r
#Graphical Representation (Point Graph)
edx_Max_Crimes_Borough_Year %>%mutate(boroughMaxYearCrime = paste0(borough,"-","(",Year,")"))%>%
  ggplot(aes(reorder(boroughMaxYearCrime,CrimeCount),CrimeCount))+
  geom_point()+
  scale_y_comma()+
  coord_flip()+
  geom_text(aes(label=comma(CrimeCount)),hjust=1)+
  theme_classic()+
  theme(legend.position = "none")+
  labs(title = "Max Crimes for each Borough",x="Borough and year of max Crimes ",y="Crime Count")
```



The visualization shows different Boroughs having the maximum crime count and the corresponding year when it has happened. Having understanding the trend, the year with Boroughs having maximum highest crime count can be determined as below:

```r
edx_Boroughs_with_max_incidents <- edx_london_crimes %>%
  group_by(borough,Year = year) %>%
  summarise(CrimeCount=sum(value)) %>%
  ungroup() %>%
  group_by(borough) %>%
  filter(CrimeCount==max(CrimeCount)) %>%
  ungroup() %>%
  count(Year,sort = TRUE,name = "Boroughs_with_max_incidents")%>%
  arrange(desc(Boroughs_with_max_incidents))
datatable(edx_Boroughs_with_max_incidents, rownames = FALSE, filter="top", options = list(pageLength = 5
```

Show 50 entries Search:

Year	Boroughs_with_max_incidents
All	All
2016	10
2008	9
2012	5
2011	4
2010	3
2009	2

Showing 1 to 6 of 6 entries Previous 1 Next

The above tabular representation above depicts 2016 has maximum number of Boroughs with highest crime count spannin 2008-2016.

After undertanding the data pattern with respect to Crime count and its segreggation across Boroughs spanning the years 2008 -2016, it is required to do a deep down analysis with respect to the other two data features - Major and Minor Category. The expectation is to understand how the categories have attributed to Crime Count in the London Crime Data Set.

```
#Tabular Representation
edx_Maj_Cat_Crimes <- edx_london_crimes %>%
  group_by(major_category) %>%
  summarize(CrimeCount = n()) %>%
  arrange(desc(CrimeCount)) %>%
  ungroup()

datatable(edx_Maj_Cat_Crimes, rownames = FALSE, filter="top", options = list(pageLength = 50, scrollX=T)
```

Show 50 entries   Search:

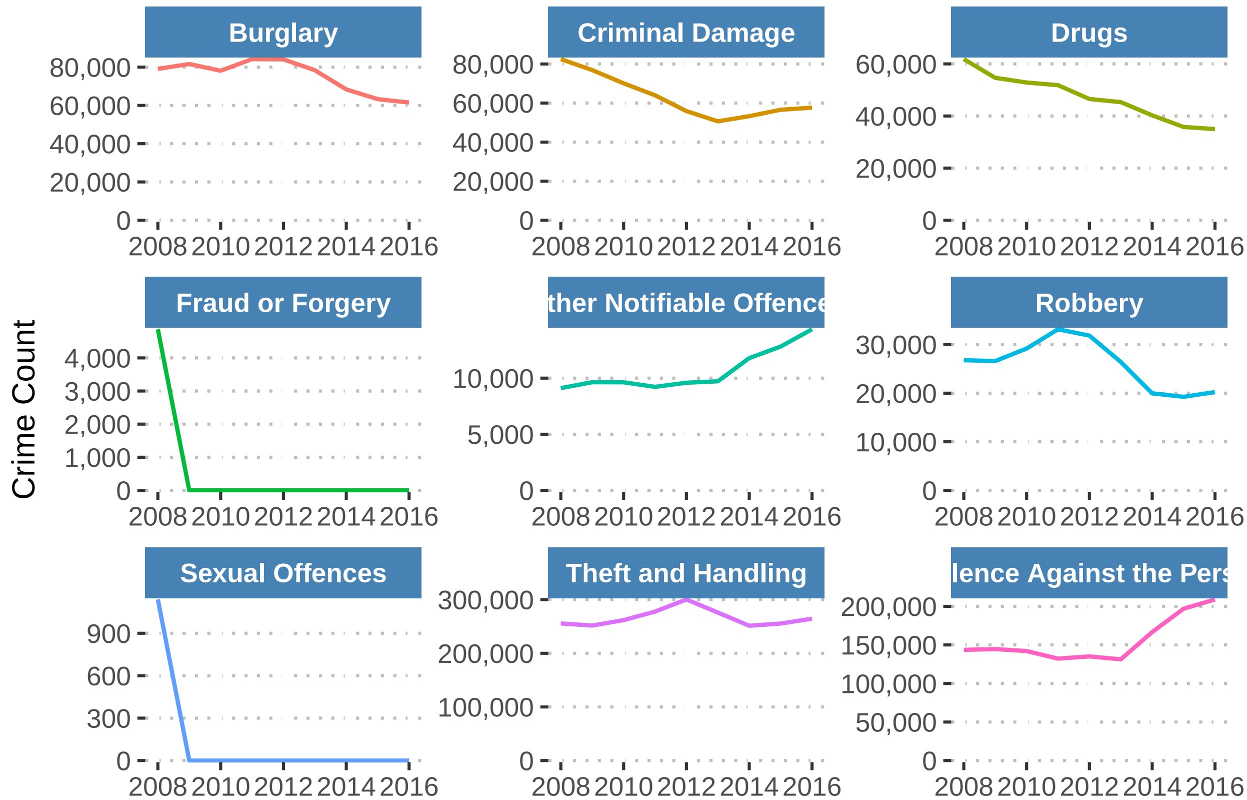
major_category	CrimeCount
Theft and Handling	3569388
Violence Against the Person	2854488
Criminal Damage	1861685
Drugs	1061859
Burglary	939239
Robbery	845979
Other Notifiable Offences	698736
Fraud or Forgery	213027
Sexual Offences	97142

Showing 1 to 9 of 9 entries   Previous 1 Next

The tabular representation shows the Major Category sorted in descending order with respect to the corresponding Crime Count. “Theft and Handling” and “Violence Against the Person” proves to be major contributor in the over all crime count from 2008-2016.

The next visual representation is to understand how Crime Count has behaved over the years spanning 2008 -2016 for each of the Major Category.

```
#Graphical Representation (Yearly Timeline) (Line Graph)
edx_london_crimes %>%
  group_by(Yearly=year,major_category) %>%
  summarise(CrimeCount=sum(value,na.rm = TRUE)) %>%
  ggplot(aes(Yearly,CrimeCount))+
  geom_line(aes(color=major_category),size=0.75)+
  theme_pubclean()+
  scale_y_comma()+
  expand_limits(y=0)+
  facet_wrap(~major_category,scales = "free")+
  labs(y="Crime Count",x=" ")+
  theme(legend.position = "none",strip.background = element_rect(fill="steelblue"),strip.text=element_t
```



The Yearly Time line graph clearly depicts that for "Theft and Handling" and "Violence Agianst the person", there is steady increase in count from 2014 -2016. "Fraud or Forgery" and "Sexual Offences" have flattend after 2008, one of the probable reason being the both of them are now reported against the category "Violence Against the Person". The relationship and impact between Major and Minor Category needs to be visualized and explored as well.

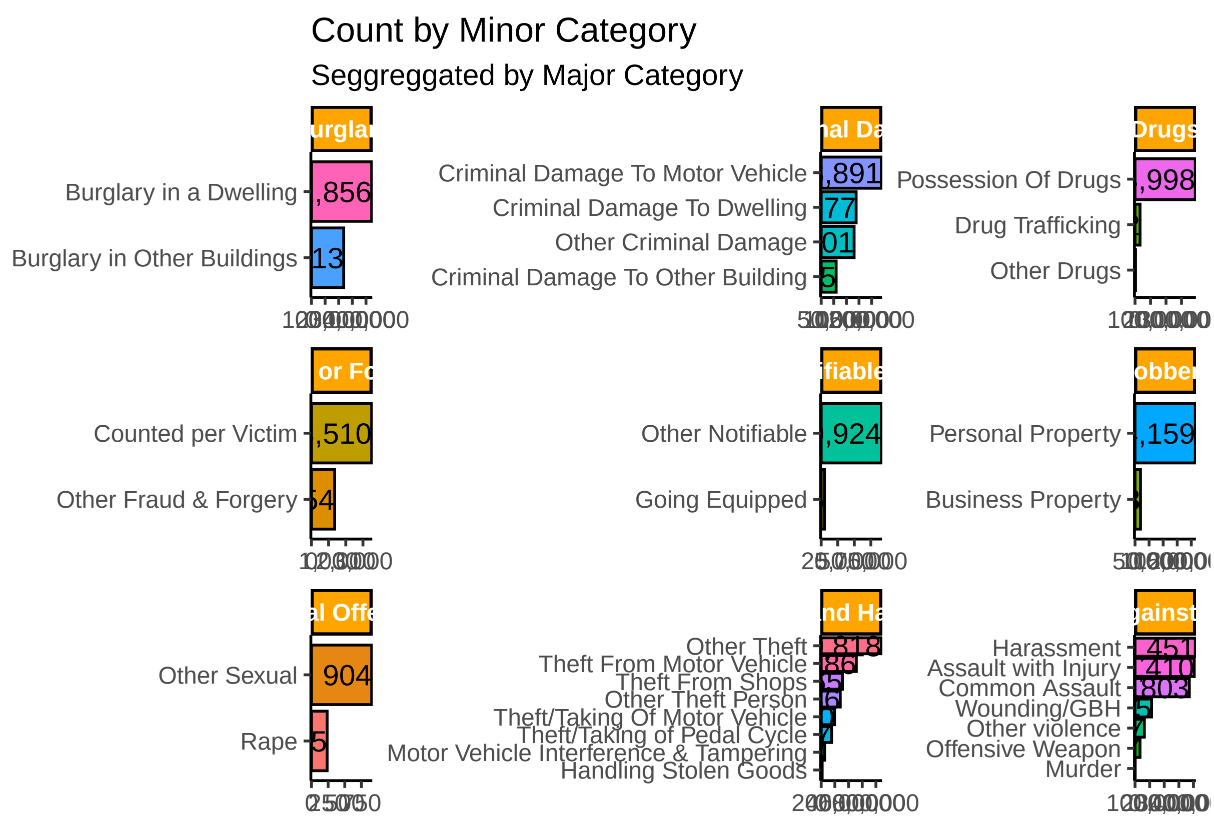
```
edx_london_crimes_Maj_Min_Category <-edx_london_crimes %>%
  group_by(major_category,minor_category) %>%
  summarise(CrimeCount=sum(value)) %>%
  arrange(desc(CrimeCount)) %>%
  ungroup()

#Tabular Representation
datatable(edx_london_crimes_Maj_Min_Category, rownames = FALSE, filter="top", options = list(pageLength
```

Show 50 entries Search:

major_category	minor_category	CrimeCount
All	All	All
Theft and Handling	Other Theft	879818
Theft and Handling	Theft From Motor Vehicle	513186
Burglary	Burglary in a Dwelling	441856
Violence Against the Person	Harassment	412451
Violence Against the Person	Assault with Injury	405410
Drugs	Possession Of Drugs	388998
Violence Against the Person	Common Assault	371803
Theft and Handling	Theft From Shops	310365
Theft and Handling	Other Theft Person	278686
Criminal Damage	Criminal Damage To Motor Vehicle	238891
Burglary	Burglary in Other Buildings	236813
Robbery	Personal Property	214159
Theft and Handling	Theft/Taking Of Motor Vehicle	195330
Theft and Handling	Theft/Taking of Pedal Cycle	151857
Criminal Damage	Criminal Damage To Dwelling	138577
Criminal Damage	Other Criminal Damage	130601
Violence Against the Person	Wounding/GBH	112835
Other Notifiable Offences	Other Notifiable	90924
Violence Against the Person	Other violence	63757
Criminal Damage	Criminal Damage To Other Building	59355
Theft and Handling	Motor Vehicle Interference & Tampering	50543
Violence Against the Person	Offensive Weapon	34227
Drugs	Drug Trafficking	32322
Robbery	Business Property	19263
Theft and Handling	Handling Stolen Goods	14499
Other Notifiable Offences	Going Equipped	4936
Fraud or Forgery	Counted per Victim	3510
Drugs	Other Drugs	2709
Fraud or Forgery	Other Fraud & Forgery	1354
Sexual Offences	Other Sexual	904
Violence Against the Person	Murder	853
Sexual Offences	Rape	235

Showing 1 to 32 of 32 entries Previous 1 Next

```r
#Graphical Representation (Bar Graph)
edx_london_crimes_Maj_Min_Category%>%
  mutate(minor_category=reorder_within(minor_category,CrimeCount,major_category)) %>%
  ggplot(aes(minor_category,CrimeCount))+
  geom_bar(aes(fill=minor_category),stat = "identity",color="black")+
  coord_flip()+
  scale_x_reordered()+
  scale_y_comma()+
  geom_text(aes(label=comma(CrimeCount)),hjust=1)+
  theme_classic()+
  theme(legend.position = "none")+
  facet_wrap(~major_category,scales ="free")+
  labs(x=" ",y=" ",title = "Count by Minor Category",subtitle = "Seggreggated by Major Category")+
  theme(legend.position = "none",strip.background = element_rect(fill="orange"),strip.text=element_text
```



As per the above representation, the Minor Categories with the highest Crime Count belong to "Theft and Handling" and "Violence Against the Person" Major Category.

Conclusions: 1. 2014 has least Crime Count; the count is increasing henceforth from 2014-2016. 2. Westminster has the highest Crime Count and London has the lowest Crime count. 3. London has the highest percentage increase in Crime from 2008 to 2016. 4. Crime Count in Westminster is steady and there is no steep increase in count in the span 2008-2016. 5. 2016 has the maximum number of Boroughs with highest Crime count. 6. Theft and Handling" and "Violence Against the Person" proves to be major contributor in the over all crime count from 2008-2016. 7. Theft and Handling" and "Violence Agianst the person", there is steady increase in count from 2014 -2016. 8."Fraud or Forgery" and "Sexual Offences" have flattend after 2008

IV. Machine Learning Algorithm for Prediction

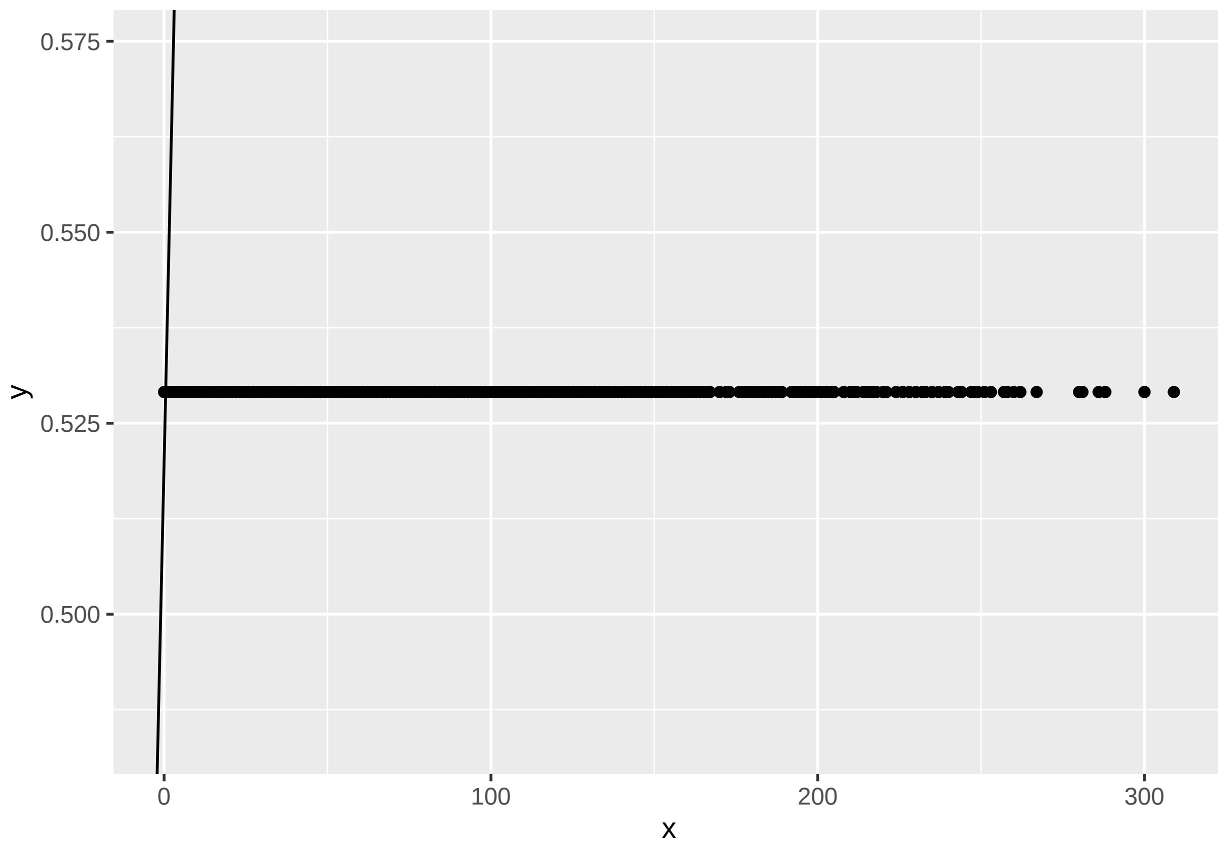
The idea is to understand and find relationship in order to predict a model to access the Crime Count corresponding to the Major Category - Theft and Handling" and "Violence Against the Person" based on the available data in Lodon Crime Dataset in 2008-2016. The following algorithms were selected for generating the prediction model and fitting it against the Validation data set (generated in the Data Loading Section). 1.Linear Regression 2.PENALIZED LEAST SQUARE for Root Mean Square Error (RMSE) calculation 3.K-nearest neighbour (KNN) 4. Random Forest

#Linear Regression

```
edx_Maj_Cat <- as.numeric(edx_london_crimes$major_category %in% c("Theft and Handling","Violence Agains

lm_fit_major_category <- mutate(edx_london_crimes, y = edx_Maj_Cat) %>% lm(y~ value, data = .)
p_hat_major_catagory <- predict(lm_fit_major_category, validation)

summary(lm_fit_major_category)
```

```
##
## Call:
## lm(formula = y ~ value, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4393 -0.5199  0.4418  0.4801  0.4801
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.199e-01  1.480e-04  3511.8   <2e-16 ***
## value       1.916e-02  8.087e-05   236.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.498 on 12141541 degrees of freedom
## Multiple R-squared:  0.0046, Adjusted R-squared:  0.0046
## F-statistic: 5.611e+04 on 1 and 12141541 DF,  p-value: < 2.2e-16
```

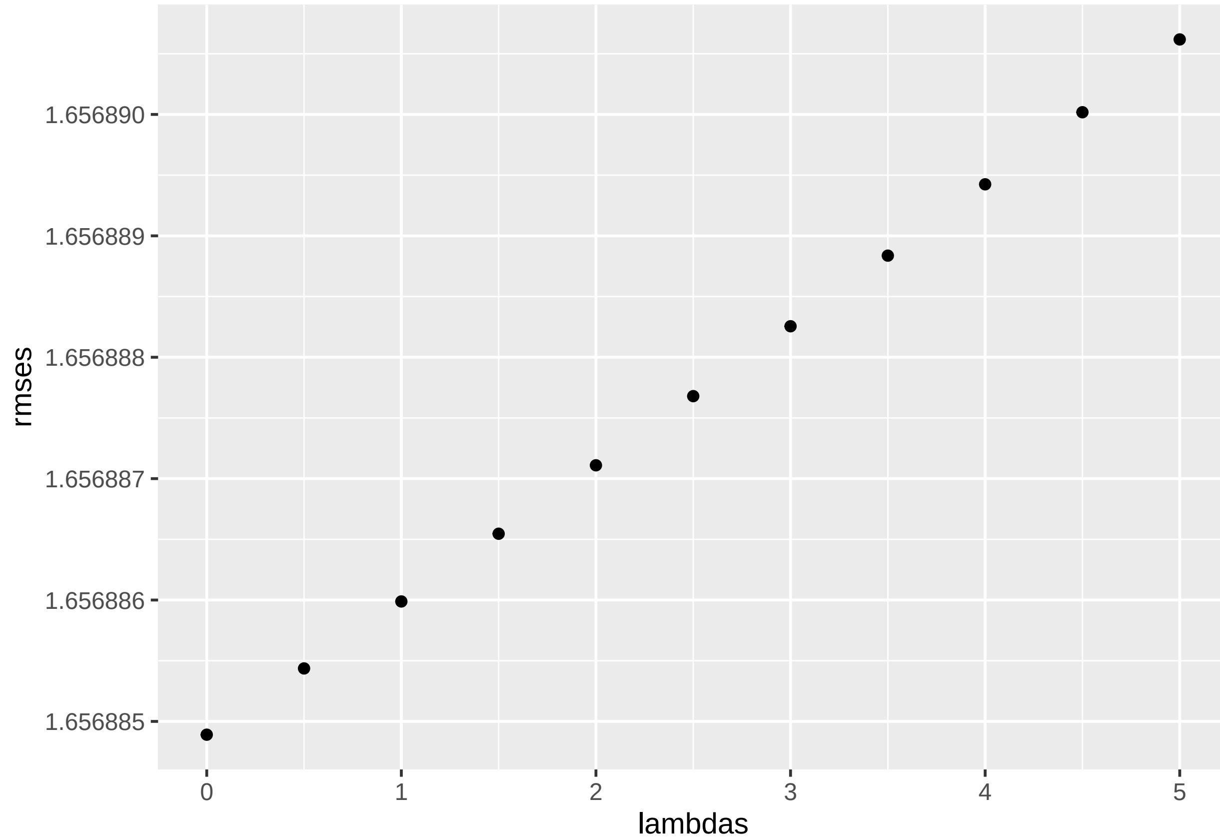
```
#Coefficients
coefs <- tidy(lm_fit_major_category, conf.int = TRUE)

#Graphical Representation
edx_london_crimes %>%
  mutate(x = value) %>%
  group_by(x) %>%
  summarize(y = mean(edx_Maj_Cat)) %>%
  ggplot(aes(x, y)) +
  geom_point() +
  geom_abline(intercept = lm_fit_major_category$coef[1], slope = lm_fit_major_category$coef[2])
```



The model seems to be linear based on the calculation and the corresponding graphical relationship above. #Root Mean Square Error (RMSE)

```
MSE <- function(true_count, predicted_count){
  sqrt(mean((true_count - predicted_count)^2))
}

#Choose Lambda Values for tuning
lambdas <- seq(0,5,.5)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_london_crimes$value)

  b_i <- edx_london_crimes %>%
    group_by(lsoa_code) %>%
    summarize(b_i = sum(value - mu)/(n() + l))

  b_u <- edx_london_crimes %>%
    left_join(b_i, by='lsoa_code') %>%
    group_by(major_category) %>%
    summarize(b_u = sum(value - b_i - mu)/(n() +l))

  predicted_count <- edx_london_crimes %>%
    left_join(b_i, by = "lsoa_code") %>%
    left_join(b_u, by = "major_category") %>%
    mutate(pred = mu + b_i +  b_u) %>% .$pred
```

```r
  return(RMSE(predicted_count, edx_london_crimes$value))
})

qplot(lambdas, rmses)
```




```r
lambda <- lambdas[which.min(rmses)]
paste('Optimal RMSE of',min(rmses),'is achieved with Lambda',lambda)
```

```
## [1] "Optimal RMSE of 1.65688489081624 is achieved with Lambda 0"
```

```r
# Predicting the validation set

mu <- mean(validation$value)
l <- lambda
b_i <- validation %>%
  group_by(lsoa_code) %>%
  summarize(b_i = sum(value - mu)/(n() + l))

b_u <- validation %>%
  left_join(b_i, by='lsoa_code') %>%
  group_by(major_category) %>%
  summarize(b_u = sum(value - b_i - mu)/(n() +l))

predicted_count <- validation %>%
```

```
  left_join(b_i, by = "lsoa_code") %>%
  left_join(b_u, by = "major_category") %>%
  mutate(pred = mu + b_i +  b_u) %>% .$pred


RMSE(predicted_count, validation$value)
```

```
## [1] 1.692983
```

The RMSE calculated and validated against the Validation set - 1.6569

#K Nearest Neighbour (KNN) The London DataSet is big (889M);As a result, there has been instances with R where "Memory Leak" has been thrown. Moreover, there was error related to KNN calculation because of the big data set. Hence, the Training and Validation data set are re-generated on a more filter data set. Also a small optimal amount of "noise" has been added in order to deal with the large data set and trick through the errors.

```
#Create Train and Test Data Set



edx_Knn_london_crimes <- london_crimes %>%
  filter(major_category %in% c("Theft and Handling","Violence Against the Person")) %>% group_by(year,ma
  summarise(crimeCount = sum(value)) %>% ungroup()

#Adding noise to handle large data set
x <- edx_Knn_london_crimes$crimeCount
corrupt <- rbinom(length(x),1,0.4)    # choose an average of 40% to corrupt at random
corrupt <- as.logical(corrupt)
noise <- rnorm(sum(corrupt),1000,200) # generate the noise to add
edx_Knn_london_crimes$crimeCount[corrupt] <- x[corrupt] + noise

test_index <- createDataPartition(edx_Knn_london_crimes$major_category, times = 1, p = 0.3, list = FALSE
test_crime_set <- as.data.frame(edx_Knn_london_crimes[test_index, ])
train_crime_set <- as.data.frame(edx_Knn_london_crimes[-test_index, ])

ks <- seq(9, 27,3)
F_1 <- sapply(ks, function(k){
knn_fit <- knn3(as.factor(major_category) ~ as.numeric(crimeCount), data = train_crime_set, k = k, use.a
y_hat <- predict(knn_fit, test_crime_set, type = "class") %>%
factor(levels = levels(as.factor(train_crime_set$major_category)))
F_meas(data = y_hat, reference = as.factor(test_crime_set$major_category))
})
```

```
## Warning in knn3Train(train = structure(c(283692, 159844, 157894,
## 310366.026986946, : k = 15 exceeds number 12 of patterns

## Warning in knn3Train(train = structure(c(283692, 159844, 157894,
## 310366.026986946, : k = 18 exceeds number 12 of patterns

## Warning in knn3Train(train = structure(c(283692, 159844, 157894,
## 310366.026986946, : k = 21 exceeds number 12 of patterns
```

```
## Warning in knn3Train(train = structure(c(283692, 159844, 157894,
## 310366.026986946, : k = 24 exceeds number 12 of patterns

## Warning in knn3Train(train = structure(c(283692, 159844, 157894,
## 310366.026986946, : k = 27 exceeds number 12 of patterns
```

```
Best_Fit <- max(F_1)
Best_K <- ks[which.max(F_1)]
Best_Fit
```

```
## [1] 1
```

```
Best_K
```

```
## [1] 9
```

The best fitof the prediction model as per the KNN method is 0.8571429.

#Random Forest

The same Training and Validation set as in KNN calculation has been used for Random Forest algorithm

```
train_rf <- randomForest(as.factor(major_category) ~ ., data=train_crime_set)
confusionMatrix(predict(train_rf, test_crime_set), as.factor(test_crime_set$major_category))$overall["A
```

```
## Accuracy
##        1
```

The Accuracy of the prediction using Random Forest is 0.68.

V. Conclusion

1. It has been observed though the Major Category like "Theft and Handling" and "Violence Against the Person" etc. have a linear relation ship with the Crime Count; but the linear regression model has not proved to be very effective and accurate in predicting the crime count for these highly attributed Major Categories. 2.Machine learning algorithm KNN (K-nearest neightbour) and Random Forest have provided a better fit of the prediction model.
2. Data visualization and exploration is quite immportant in the context of Crime Count prediction.