Analysis of Los Angeles Crime Data

Final Report

AIT 664 - Information: Representation, Processing and Visualization

Dr. Charles Lynch

Praneeth Ravirala (G01448129)

Shalvi Sanjay Lale (G01419005)

Vivek Patil Paidigumal (G01450948)

GEORGE MASON UNIVERSITY

1st December 2024

Analysis of Los Angeles Crime Data

INTRODUCTION

This project explores crime patterns across Los Angeles by integrating crime data with information from businesses and schools. The goal is to identify high, medium, and low crime zones, assess crime trends over time, and evaluate the impact on essential services and community well-being. Insights from the analysis will help the government enhance security, guide the public in making safer living choices, and assist businesses in selecting favorable locations. By understanding when and where crimes occur, the research aims to offer recommendations to reduce criminal activity, improve public safety, and foster sustainable economic growth.

DATA ACQUISITION

Initial Requirements

- Languages: Python programming language
- Software: Jupyter Notebook, Tableau, MS-Excel
- Libraries: Pandas, Numpy, Matplotlib, Seaborn, Scikit-Learn
- Hardware: 4GB RAM, 4GB HardDisk
- Operating System: Windows, Linux, MacOS
- Processor: Intel, AMD, Apple Silicon

Data Sources

- US Government Open Data
- Los Angeles County Open Data

Data Necessary for inputs to the analysis

- Date and Time of the crime when the crime has occurred
- Age and Gender of the victims involved in the crime
- Derived Categorised Neighbourhood based on Neighbourhoods retrieved using Coordinates of Each Crime Location
- Number of Schools, Crimes, Businesses and Mean School Enrollment in each categorised Neighbourhood

Research question and it's hypothesis

1. **Research Question:** How crimes in particular time frames can benefit the government in regularizing the policies, promoting safety of the public and enhancing businesses.
   Hypothesis: Retrieving Number of Crimes in various time frames helps in understanding and recommending the government to enhance their policies accordingly, promoting public and economic protection.
   Rationale: Calculate Number of Crimes and using visualization to plot the Crimes with time frames helps us to know which time intervals are more prone to crimes.

2. **Research Question:** How does categorizing the areas to different neighbourhoods help the government in organizing better security infrastructure in high crime areas, helping the public to be aware from being victimized and promoting business establishments.
   Hypothesis: By understanding the concentration of crimes in different areas helps the government to enhance the security infrastructure wherever required in return protecting the people and businesses.
   Rationale: Using the coordinates of the crime locations helps in visualizing the dispersion of crimes to different neighbourhoods to understand the concentration crimes.

3. **Research Question:** How crimes are affecting establishments of essential facilities like schools and Businesses in different Neigbourhoods, which changes the preferences of public and businesses to locate in these areas.
   Hypothesis: If crime rates are high in a few areas, it might help people and businesses in deciding if they wish to locate in that area.
   Rationale: Finding the relationship between Mean Enrollment of schools,businesses with that of the number of crimes in that area helps in understanding the impact of crime on beneficiary preferences.

DATA PREPARATION/PREPROCESSING

- Importing Datasets: Importing datasets using pandas lib followed by the data sampling for faster processing

```
[220]: import pandas as pd
       from datetime import datetime
       import requests
       import numpy as np
```

```
[221]: import nltk
       from nltk.tokenize import word_tokenize
```

```
[222]: fp=pd.read_csv('/Users/praneethravirala/Downloads/Crime_Data_from_2020_to_Present.csv')
       fp1=pd.read_csv('/Users/praneethravirala/Downloads/Crime_Data_from_2020_to_Present.csv')
       fp2=fp2=pd.read_excel('/Users/praneethravirala/Documents/Schools_LosAngeles.xlsx')
```

Fig. Importing Datasets
Tool - Jupyter ; **Language** - Python

Data Sampling: For faster Computation, we take samples of three datasets which contain 10000 rows for our analysis using pandas sample function().

```
[10]: fp_sample=fp.sample(n=10000)
      fp_sample.head(10)
```

[10]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	L
230484	210313394	7/22/21 0:00	7/22/21 0:00	1500	3	Southwest	331	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	...	IC	Invest Cont	341.0	NaN	NaN	NaN	EX
481780	221213635	6/11/22 0:00	6/10/22 0:00	1545	12	77th Street	1267	2	930	CRIMINAL THREATS - NO WEAPON DISPLAYED	...	AO	Adult Other	930.0	NaN	NaN	NaN	60
275432	210107638	3/18/21 0:00	3/18/21 0:00	1500	1	Central	142	1	510	VEHICLE - STOLEN	...	IC	Invest Cont	510.0	NaN	NaN	NaN	400
681546	230717767	11/29/23 0:00	11/29/23 0:00	2015	7	Wilshire	787	2	624	BATTERY - SIMPLE ASSAULT	...	AA	Adult Arrest	624.0	NaN	NaN	NaN	WAS
		9/3/21	9/3/21							ASSAULT WITH DEADLY			Adult					

Fig. Data Sampling
Tool - Jupyter ; **Language** - Python

- Structuring Data as per the Required Format: Structuring Time Column to the required format ‘HH:MM’ using pandas datetime() and loc() function.

```
[225]: fp['TIME OCC']=fp['TIME OCC'].astype(str)

[186]: for i in fp.index:
           fp.loc[i,'TIME OCC']=fp.loc[i,'TIME OCC'].zfill(4)
           fp.loc[i,'TIME OCC']=datetime.strptime(fp.loc[i,'TIME OCC'],"%H%M").strftime("%H:%M")

[187]: fp['TIME OCC']

[187]: 0        21:30
       1        18:00
       2        17:00
       3        20:37
       4        12:00
                ...
       99995    15:50
       99996    17:22
       99997    21:30
       99998    01:00
       99999    20:00
       Name: TIME OCC, Length: 100000, dtype: object
```

Fig. Structuring Time column into HH:MM format
Tool - Jupyter ; **Language** - Python

- Structuring ‘DATE OCC’ column into the required format ie. ‘YYYY-MM-DD’ using pandas datetime() and loc() function.

```
[195]: for i in fp.index:
           l=nltk.word_tokenize(fp.loc[i,'DATE OCC'])
           text=l[1]
           fp.loc[i,'DATE OCC']=fp.loc[i,'DATE OCC'].replace(text,"")
```

Fig. Structuring Date Occurred column
Tool - Jupyter ; **Language** - Python

```
[193]: fp['DATE OCC']=fp['DATE OCC'].astype(str)

[145]: fp['Date Rptd']=fp['Date Rptd'].astype(str)

[196]: for i in fp.index:
           fp.loc[i,'DATE OCC']=datetime.strptime(fp.loc[i,'DATE OCC'].strip(),'%m/%d/%y').date()
```

Extracting Date in Specified format
Tool - Jupyter ; **Language** - Python

```
[197]: fp['DATE OCC']
```

```
[197]: 0        2020-03-01
       1        2020-02-08
       2        2020-11-04
       3        2020-03-10
       4        2020-08-17
                   ...
       99995    2020-04-04
       99996    2020-02-14
       99997    2020-07-03
       99998    2020-12-03
       99999    2020-03-06
       Name: DATE OCC, Length: 100000, dtype: object
```

Fig. Date Occurred column in YYYY-MM-DD Format
Tool - Jupyter ; **Language** - Python

- Correcting Data Types: Converting 'Vict Age' column from float to integer using astype() function.

```
[201]: fp['Vict Age']=fp['Vict Age'].astype(int)
       fp['Vict Age']
```

```
[201]: 0         0
       1        47
       2        19
       3        19
       4        28
                ..
       99995    63
       99996    26
       99997    40
       99998     0
       99999     0
       Name: Vict Age, Length: 100000, dtype: int64
```

Fig. Changing Victim age column from float to integer
Tool - Jupyter ; **Language** - Python

- Geo-Coding: Retrieving 'neighborhood' values by using google geo-coding API based on coordinates (latitude and longitude)

  We apply the function to coordinates on all the three datasets

```
[3]: def get_address_from_lat_lng(latitude, longitude):
         api_key = 'AIzaSyASnSSrrQpqSIjgcUeuYKftPgkhElj3RS8'
         url = f'https://maps.googleapis.com/maps/api/geocode/json?latlng={latitude},{longitude}&key={api_key}'
         response = requests.get(url)
         data = response.json()
         if data['status'] == 'OK':
             for component in data['results'][0]['address_components']:
                 if 'neighborhood' in component['types']:
                     return component['long_name']
         return None
```

Fig. Defining Function which take Latitude and Longitude with API key
Tool - Jupyter ; **Language** - Python

Using Tuple of Coordinates, Iterating the tuples through a loop we retrieve the neighbourhood of each tuple by calling api() function.

```
•[178…  lat=fp['LAT']
        lon=fp['LON']
        co=list(zip(lat,lon))
        count=0
        l1=[]
        for i,j in co:
            if count<=10000:
                address = get_address_from_lat_lng(i,j)
                if address:
                    l1.append(address)
                else:
                    l1.append('NA')
                count=count+1
```

Fig. Calling the API function and retrieving the address components(neighborhood) into a list
Tool - Jupyter ; **Language** - Python

Similarly we retrieve neighbourhoods for all the three datasets which are used for data integration.

```
[176]: fp['Neighbourhood']

[176]: 0                   Mid-City
       1       Downtown Los Angeles
       2          South Los Angeles
       3               Sherman Oaks
       4                 Central LA
                        ...
       9995                Tarzana
       9996              Sun Valley
       9997       South Los Angeles
       9998       Pacific Palisades
       9999       South Los Angeles
       Name: Neighbourhood, Length: 10000, dtype: object
```

Fig. The list of address values(neighborhood) retrieved by API call
Tool - Jupyter ; **Language** - Python

DATA CLEANING

- Removing Duplicates: We have removed duplicate values by ignoring index values in all three datasets by using drop_duplicates() function from Pandas

```
[10]: fp_sample=fp_sample.drop_duplicates(ignore_index=True)
      fp_sample
```

[10]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION	Cross Street
0	240905280	2/5/24 0:00	2/5/24 0:00	1200	9	Van Nuys	971	1	440	THEFT PLAIN - PETTY ($950 & UNDER)	...	IC	Invest Cont	440.0	NaN	NaN	NaN	15300 VENTURA BL	NaN
1	231316913	9/7/23 0:00	9/3/23 0:00	1600	13	Newton	1323	1	310	BURGLARY	...	IC	Invest Cont	310.0	NaN	NaN	NaN	700 E 27TH ST	NaN
2	200111453	5/7/20 0:00	5/7/20 0:00	1230	1	Central	192	1	310	BURGLARY	...	AO	Adult Other	310.0	998.0	NaN	NaN	1600 S HOPE ST	NaN
3	230612334	7/10/23 0:00	7/10/23 0:00	935	6	Hollywood	615	1	310	BURGLARY	...	IC	Invest Cont	310.0	NaN	NaN	NaN	3200 LEDGEWOOD DR	NaN
4	231701226	9/21/23 0:00	9/21/23 0:00	1750	17	Devonshire	1725	1	310	BURGLARY	...	IC	Invest Cont	310.0	998.0	NaN	NaN	19600 TULSA ST	NaN

Fig. Remove Duplicates from Crime Dataset
Tool - Jupyter ; **Language** - Python

```
fp1_sample=fp1_sample.drop_duplicates(ignore_index=True)
fp1_sample
```

:

	LOCATION ACCOUNT #	BUSINESS NAME	DBA NAME	STREET ADDRESS	CITY	ZIP CODE	LOCATION DESCRIPTION	MAILING ADDRESS	MAILING CITY	MAILING ZIP CODE	NAICS	PRIMARY NAICS DESCRIPTION	COUNCI DISTRIC
0	0002848652-0001-1	409 N GENESEE LP	NaN	409 N GENESEE AVENUE	LOS ANGELES	90036-2215	409 GENESEE 90036-2215	135 S LA BREA AVENUE APT #6	LOS ANGELES	90036-2900	531100.0	Lessors of real estate (including mini warehou...	
1	0003106131-0001-5	JAMES COSTOS INC	NaN	595 S MAPLETON DRIVE	LOS ANGELES	90024-1810	595 MAPLETON 90024-1810	10866 WILSHIRE BLVD SUITE #900	LOS ANGELES	90024-4352	NaN	NaN	
2	0003218522-0001-5	VALENTINE VALDOVINOS	A SAFE SPACE-THERAPY SERVICES	1800 EL CERRITO PLACE UNIT #16	LOS ANGELES	90068-3744	1800 EL CERRITO 90068-3744	3948 VAN HORNE AVENUE	LOS ANGELES	90032-1144	621330.0	Offices of mental health practitioners (except...	

Fig. Remove Duplicates from Business Dataset
Tool - Jupyter ; **Language** - Python

```
[17]: fp2=pd.read_excel('/Users/praneethravirala/Documents/Schools_LosAngeles.xlsx')
      fp2=fp2.drop_duplicates(ignore_index=True)
      fp2
```

[17]:

	ObjectID	Category1	Category2	Category3	Name	Label	Address Line 1	Address Line 2	City	State	...	Organization	
0	2	Education	Elementary Schools	Charter Schools	Jardin de la Infancia	Jardin de la Infancia	1400 S Broadway Los Angeles, CA 90015	NaN	Los Angeles	CA	...	Los Angeles County Office of Education	California Dep Education (CI
1	12	Education	Intermediate/Middle/Junior High Schools	Charter Schools	Russell Westbrook Why Not? Middle	Russell Westbrook Why Not? Middle	1700 West 46th Street	NaN	Los Angeles	CA	...	Los Angeles County Office of Education	California Dep Education (CI
2	14	Education	High Schools	Charter Schools	Russell Westbrook Why Not? High	Russell Westbrook Why Not? High	1755 West 52nd Street	NaN	Los Angeles	CA	...	Los Angeles County Office of Education	California Dep Education (CI
3	19	Education	Elementary Schools	Charter Schools	Lashon Academy City	Lashon Academy City	3109 Sixth Avenue	NaN	Los Angeles	CA	...	Los Angeles County Office of Education	California Dep Education (CI
4	23	Education	Elementary Schools	Charter Schools	KIPP Poder Public	KIPP Poder Public	630 Leonard Avenue	NaN	Los Angeles	CA	...	Los Angeles County Office of	California Dep Education (CI

Fig. Remove Duplicates from Schools Dataset
Tool - Jupyter ; **Language** - Python

- Checking Outliers: Retrieved statistical information of victim age using describe function to get mean, median, min, max and quartiles of the variable. We have found some outliers after visualization in the box plot and we have tried to treat them using interquartile range.

```
[38]: f['Vict Age'].describe()

[38]: count    20291.000000
      mean        35.130895
      std         13.709383
      min          2.000000
      25%         28.000000
      50%         29.000000
      75%         42.000000
      max         99.000000
      Name: Vict Age, dtype: float64
```

Fig. Retrieving Summary Statistics of Vict Age
Tool - Jupyter ; **Language** - Python

```
[204]: plt.boxplot(fp['Vict Age'])
```

```
[204]: {'whiskers': [<matplotlib.lines.Line2D at 0x1441120f0>,
         <matplotlib.lines.Line2D at 0x1440a1400>],
        'caps': [<matplotlib.lines.Line2D at 0x1441138f0>,
         <matplotlib.lines.Line2D at 0x144111730>],
        'boxes': [<matplotlib.lines.Line2D at 0x144111f10>],
        'medians': [<matplotlib.lines.Line2D at 0x144110f80>],
        'fliers': [<matplotlib.lines.Line2D at 0x144110ce0>],
        'means': []}
```

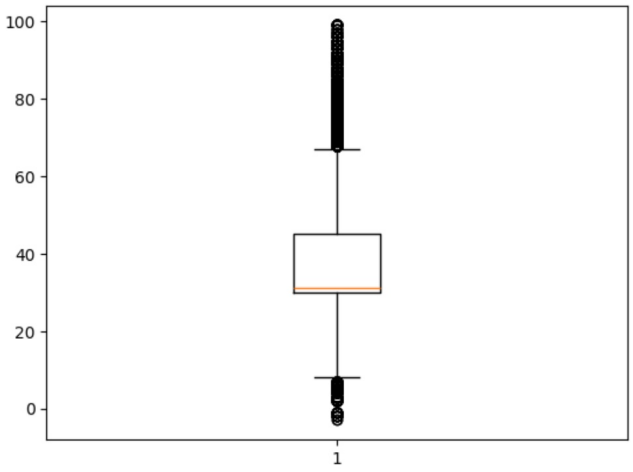

Fig. To check the outliers of victim age so that we can treat them for better model performance
Tool - Jupyter ; **Language** - Python

```
[41]: Q3=42
      Q1=28
      IQR=Q3-Q1
      max1=Q3+1.5*IQR
      min1=Q1-1.5*IQR
      for i in f.index:
          if f.loc[i,'Vict Age']>max1:
              f.loc[i,'Vict Age']=max1
          elif f.loc[i,'Vict Age']<min1:
              f.loc[i,'Vict Age']=min1
```

```
[42]: f['Vict Age'].describe()
```

```
[42]: count    20291.000000
      mean        34.735301
      std         12.581158
      min          7.000000
      25%         28.000000
      50%         29.000000
      75%         42.000000
      max         63.000000
      Name: Vict Age, dtype: float64
```

Fig. Treating Outliers using Interquartile Range
Tool - Jupyter ; **Language** - Python

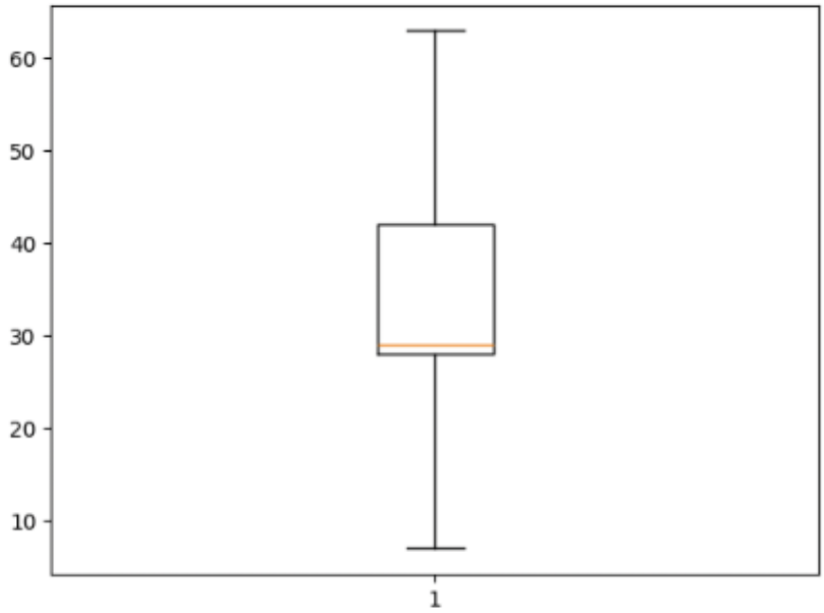

Fig. After Treating Visualising Vict Age to Check There are no outliers.
Tool - Jupyter ; **Language** - Python

- Treating Null Values

```
[210]: fp['Vict Sex'].unique()

[210]: array(['M', 'X', 'F', 'Other'], dtype=object)

[212]: fp['Vict Sex']=fp['Vict Sex'].fillna('X')

[159]: fp['Vict Descent'].unique()

[159]: array(['O', 'X', 'H', 'B', 'W', nan, 'A', 'K', 'C', 'J', 'F', 'I', 'V',
              'S', 'P', 'Z', 'G', 'U', 'D', 'L'], dtype=object)

[162]: fp['Vict Descent']=fp['Vict Descent'].fillna('X')

fp1['LOCATION']=fp1['LOCATION'].fillna('0,0')

7]: fp2['Enrollment']=fp2['Enrollment'].fillna(round(np.mean(fp2['Enrollment'])))
```

Fig. We treat null values by imputing mean or median or we will keep the value as another category
Tool - Jupyter ; **Language** - Python

- Removing Unnecessary Columns: We have Dropped Unnecessary columns for our analysis from all the three datasets using pandas drop() function

```
[26]: columns_fp_sample=['Status','Status Desc','Weapon Used Cd','Weapon Desc','Crm Cd 2','Crm Cd 3','Crm Cd 4','Mocodes','Date Rptd','AREA']
      fp_sample=fp_sample.drop(columns=columns_fp_sample)

[55]: fp_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 230484 to 549632
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   DR_NO          10000 non-null  int64
 1   DATE OCC       10000 non-null  object
 2   TIME OCC       10000 non-null  object
 3   AREA NAME      10000 non-null  object
 4   Crm Cd Desc    10000 non-null  object
 5   Vict Age       10000 non-null  int64
 6   Vict Sex       10000 non-null  object
 7   Vict Descent   10000 non-null  object
 8   Premis Desc    9991 non-null   object
 9   LAT            10000 non-null  float64
 10  LON            10000 non-null  float64
 11  Neighbourhood  10000 non-null  object
dtypes: float64(2), int64(2), object(8)
```

Fig. Dropping Columns from Crime Dataset
Tool - Jupyter ; **Language** - Python

```
[251]: columns_business=['DBA NAME','CITY','LOCATION END DATE','NAICS','MAILING ADDRESS',
                         'MAILING CITY','MAILING ZIP CODE','PRIMARY NAICS DESCRIPTION','LOCATION DESCRIPTION','COUNCIL DISTRICT','LOCATION START DATE
       fp1_sample=fp1_sample.drop(columns=columns_business)

[252]: fp1_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 75978 to 49059
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   LOCATION ACCOUNT #  10000 non-null  object
 1   BUSINESS NAME       10000 non-null  object
 2   STREET ADDRESS      10000 non-null  object
 3   ZIP CODE            10000 non-null  object
 4   LOCATION            9434 non-null   object
 5   Year                9917 non-null   float64
 6   Date                9917 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(5)
memory usage: 625.0+ KB
```

Fig. Dropping Columns from Business Dataset
Tool - Jupyter ; **Language** - Python

```
[ ]: columns3=['Label','City','State','Source','Source ID','x','y','Label Class','Organization']
     fp2=fp2.drop(columns=columns3)
     fp2=fp2.drop(columns=['Category1'])
     fp2=fp2.drop(columns=['Address Line 2'])

[153]: fp2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 811 entries, 0 to 810
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ObjectID        811 non-null    int64
 1   Category1       811 non-null    object
 2   Category2       811 non-null    object
 3   Category3       804 non-null    object
 4   Name            811 non-null    object
 5   Label           811 non-null    object
 6   Address Line 1  811 non-null    object
 7   Address Line 2  0 non-null      float64
 8   City            811 non-null    object
 9   State           811 non-null    object
 10  ZIP Code        811 non-null    object
 11  Organization    811 non-null    object
 12  Source          811 non-null    object
```

Fig. Dropping Columns from Schools Dataset
Tool - Jupyter ; **Language** - Python

- Data Inconsistency

```
[211]: for i in fp.index:
           if fp.loc[i,'Vict Sex']=='H':
               fp.loc[i,'Vict Sex']='Other'

[213]: fp['Vict Sex'].unique()

[213]: array(['M', 'X', 'F', 'Other'], dtype=object)

[202]: mean_age=round(np.mean(fp['Vict Age']))
       for i in fp.index:
           if fp.loc[i,'Vict Age']==0:
               fp.loc[i,'Vict Age']=mean_age
```

Fig. We check the range of values and if any value seems to be inconsistent we impute the values.
Tool - Jupyter ; **Language** - Python

- Data Normalisation
  As there are a limited number of numerical features there is no requirement of normalisation, but for demonstration purposes we have done on 'Vict Age'.

```
[233]: std_age=fp['Vict Age'].std()
       mean_age=fp['Vict Age'].mean()

[234]: for i in fp.index:
           fp.loc[i,'Vict Age']=(fp.loc[i,'Vict Age']-mean_age)/std_age

/var/folders/pd/ngxc57bx2yzcdv41xvd2scn80000gn/T/ipykernel_1481/2012416308.py
ecated and will raise an error in a future version of pandas. Value '-1.38384
y cast to a compatible dtype first.
  fp.loc[i,'Vict Age']=(fp.loc[i,'Vict Age']-mean_age)/std_age

[235]: fp['Vict Age']

[235]: 0       -1.383847
       1        0.781159
       2       -0.508632
       3       -0.508632
       4       -0.094057
                  ...
       99995    1.518182
       99996   -0.186185
       99997    0.458711
       99998   -1.383847
       99999   -1.383847
       Name: Vict Age, Length: 100000, dtype: float64
```

Fig. Data Normalisation of Vict Age..
Tool - Jupyter ; **Language** - Python

- Deriving Columns: Deriving Vict Age Groups into categories such as 0-10,11-18,26-40,..,41-70 using pandas cut() function.

```
[12]: agegroups=[0,10,18,25,40,70]
      grp_labels=['0-10','11-18','19-25','26-40','41-70']
      f['Vict_AgeGrp']=pd.cut(f['Vict Age'],bins=agegroups,labels=grp_labels)

[13]: f['Vict_AgeGrp']

[13]: 0        26-40
      1        26-40
      2        26-40
      3        26-40
      4        26-40
               ...
      20286    26-40
      20287    26-40
      20288    26-40
      20289    26-40
      20290    26-40
      Name: Vict_AgeGrp, Length: 20291, dtype: category
      Categories (5, object): ['0-10' < '11-18' < '19-25' < '26-40' < '41-70']
```

Fig. Derived Vict Age Group Column from Vict Age.
Tool - Jupyter ; **Language** - Python

We have Derived Year Column for all the three datasets to retrieve data summary of number of schools, businesses, crimes, enrollment w.r.t each year from 2020-2024.

\

```
55]: fp8['Year']=fp8['Date'].dt.year

56]: fp8['Year']

56]: 0       2020
     1       2020
     2       2020
     3       2020
     4       2020
             ...
     9995    2020
     9996    2020
     9997    2020
     9998    2020
     9999    2020
     Name: Year, Length: 10000, dtype: int32
```

Fig. Derived Year Column for Crime Dataset
Tool - Jupyter ; **Language** - Python

```
mode_value = fp1_sample['Year'].mode()[0]
fp1_sample['Year']=fp1_sample['Year'].fillna(mode_value)
fp1_sample['Year']=fp1_sample['Year'].astype(int)

fp1_sample['Year']

75978     2015
193369    2018
259055    2011
58620     1962
142850    2023
          ...
149437    2013
141347    2011
190673    2024
211936    2018
49059     1988
Name: Year, Length: 10000, dtype: int64
```

Fig. Derived Year Column for Business Dataset
Tool - Jupyter ; **Language** - Python

```
[180]: fp2['Year']=fp2['Date'].dt.year
       fp2['Year']=fp2['Year'].astype(int)
       fp2['Year']

[180]: 0      2023
       1      2023
       2      2023
       3      2023
       4      2023
              ...
       806    2022
       807    2022
       808    2022
       809    2022
       810    2022
       Name: Year, Length: 811, dtype: int64
```

Fig. Derived Year Column for Schools Dataset
Tool - Jupyter ; **Language** - Python

- Exporting Cleaned Datasets: Once all Cleaned operations are done on all the three datasets, those are exported for Integration Purpose.

```
memory usage: 76.2+ KB

[184…  fp_sample.to_csv('/Users/praneethravirala/Documents/Laptop/664_Project/CrimeData_Cleaned.csv',index=None)
       fp1_sample.to_csv('/Users/praneethravirala/Documents/Laptop/664_Project/BusinessData_Cleaned.csv',index=None)
       fp2.to_csv('/Users/praneethravirala/Documents/Laptop/664_Project/SchoolsData_Cleaned.csv',index=None)
```

Fig. Exporting Cleaned datasets back to .csv format.
Tool - Jupyter ; **Language** - Python

- Data Integration: Importing the cleaned datasets, Initially Integrating Business and Schools dataset using inner join operation on ‘ZIP Code’ followed by integrating the result with Crime Dataset using inner join operation on ‘Neighbourhood’ to form overall dataset for Exploration, Visualisation and Modelling.

```
[77]: fp1=pd.read_csv('CrimeData_Cleaned.csv')
      fp2=pd.read_csv('BusinessData_Cleaned.csv')
      fp3=pd.read_csv('SchoolsData_Cleaned.csv')

[78]: fp3['ZIP CODE']=fp3['ZIP Code']
      fp3=fp3.drop(columns=['ZIP Code'])

[79]: fp_integrated = pd.merge(fp2,fp3,on=['ZIP CODE'],how='inner')

[80]: fp_integrated.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 33 entries, 0 to 32
      Data columns (total 19 columns):
       #   Column              Non-Null Count  Dtype
      ---  ------              --------------  -----
       0   LOCATION ACCOUNT #  33 non-null     object
       1   BUSINESS NAME       33 non-null     object
       2   STREET ADDRESS      33 non-null     object
       3   ZIP CODE            33 non-null     object
       4   LOCATION            33 non-null     object
       5   Year_x              33 non-null     int64
       6   Date_x              33 non-null     object
       7   Neighbourhood_x     29 non-null     object
       8   ObjectID            33 non-null     int64
       9   Category2           33 non-null     object
       10  Category3           33 non-null     object
       11  Name                33 non-null     object
       12  Address Line 1      33 non-null     object
```

Fig. Integrating Business and Schools Dataset using pd.merge()
Tool - Jupyter ; **Language** - Python

```
[81]: fp_integrated['Year']=fp_integrated['Year_x']
      fp_integrated['Neighbourhood']=fp_integrated['Neighbourhood_x']
      fp_integrated=fp_integrated.drop(columns=['Year_x','Neighbourhood_x'])
      fp_integrated=fp_integrated.drop(columns=['Year_y','Neighbourhood_y'])

[82]: fp_integrated1 = pd.merge(fp1,fp_integrated,on=['Neighbourhood'],how='inner')

[85]: fp_integrated1
```

[85]:

	DR_NO	TIME OCC	AREA NAME	Crm Cd Desc	Vict Age	Vict Sex	Vict Descent	Premis Desc	LAT	LON	...	ObjectID	Category2
0	210313394	15:00	Southwest	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	28	M	O	STREET	34.0246	-118.3520	...	243	Intermediate/Middle/Junior High Schools
1	210313394	15:00	Southwest	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	28	M	O	STREET	34.0246	-118.3520	...	325	Elementary Schools
2	210313394	15:00	Southwest	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	28	M	O	STREET	34.0246	-118.3520	...	1418	Elementary Schools
3	210313394	15:00	Southwest	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	28	M	O	STREET	34.0246	-118.3520	...	1618	Intermediate/Middle/Junior High Schools

Fig. Integrating Result with Crime Dataset to form Overall Integrated Dataset.
Tool - Jupyter ; **Language** - Python

- Exporting Integrated Dataset: Once we did Data Integration, we export the resultant dataset back to .csv for Analysis and Modelling

```
[86]: fp_integrated1.to_csv('Integrated_Dataset.csv',index=None)
```

Fig. Exporting Integrated Dataset back to .csv format.
Tool - Jupyter ; **Language** - Python

EXPLORATORY ANALYSIS

We have done data analysis in three stages, univariate, bivariate and multivariate.

1. In univariate analysis, we have explored victim age, victim gender and victim descent and also the time frames corresponding to the number of crimes. We have taken these variables as the statistical information and visualizations derived from these variables are helpful in answering our research questions.

Implementation

- Exploration of victim age: We have also grouped the number of crimes with respect to age groups and visualized using bar-chart to show the distribution of crimes in different age groups.

[75]:
```
tb1=f.groupby('Vict_AgeGrp')['DR_NO'].nunique().sort_index()
tb1=tb1.reset_index()
tb1=tb1.rename(columns={'DR_NO':'No of Crimes'})
tb1
```

```
/var/folders/pd/ngxc57bx2yzcdv41xvd2scn80000gn/T/ipykernel_1213/3477436
nd will be changed to True in a future version of pandas. Pass observed
default and silence this warning.
  tb1=f.groupby('Vict_AgeGrp')['DR_NO'].nunique().sort_index()
```

[75]:

	Vict_AgeGrp	No of Crimes
0	0-10	33
1	11-18	150
2	19-25	616
3	26-40	2909
4	41-70	1408

Fig. Grouping No of Crimes Based on Vict Age Group.
Tool - Jupyter ; **Language** - Python

- Exploration of victim gender: We have checked different types of gender categories involved and visualized the impact of crime on each gender. We have used both pie-chart and bar-chart for this.

```
[15]: f['Vict Sex'].unique()

[15]: array(['M', 'F', 'X'], dtype=object)

[77]: tb2=f.groupby('Vict Sex')['DR_NO'].nunique().sort_index()
      tb2=tb2.reset_index()
      tb2=tb2.rename(columns={'DR_NO':'No of Crimes'})
      tb2
```

[77]:

	Vict Sex	No of Crimes
0	F	1828
1	M	1949
2	X	1339

Fig. Grouping No of Crimes Based on Vict Sex
Tool - Jupyter ; **Language** - Python

- Exploration of victim descent: Using bar-chart we have visualized distribution of crimes w.r.t different categories of victim descent.

```
[81]: tb3=f.groupby('Vict Descent')['DR_NO'].nunique().sort_index()
      tb3=tb3.reset_index()
      tb3=tb3.rename(columns={'DR_NO':'No of Crimes'})
      tb3
```

[81]:

	Vict Descent	No of Crimes
0	A	110
1	B	958
2	C	24
3	F	18
4	H	1579
5	I	4
6	J	5
7	K	35
8	L	1

Fig. Grouping No of Crimes Based on Vict Descent
Tool - Jupyter ; **Language** - Python

- Exploration of Time Variable: We have divided the time variable into equal intervals of 2 hours i.e. 12 intervals. In each time frame we have retrieved the number of crimes and visualized using bar-chart to focus on the time frames with more crime occurrences.

```
[182]: f['TIME OCC']=pd.to_datetime(f['TIME OCC'])
tb4 = f.groupby(pd.Grouper(key='TIME OCC', freq='2H'))['DR_NO'].count()
tb4 = tb4.reset_index().rename(columns={'TIME OCC': 'Hour', 'DR_NO': 'Count'})
tb4['Hour'] = tb4['Hour'].dt.time
tb4['Hour'] = tb4['Hour'].astype(str)
tb4
```

```
/var/folders/pd/ngxc57bx2yzcdv41xvd2scn80000gn/T/ipykernel_1079/484531866.py:2: FutureWarning: 'H'
ure version, please use 'h' instead.
  tb4 = f.groupby(pd.Grouper(key='TIME OCC', freq='2H'))['DR_NO'].count()
```

[182]:

	Hour	Count
0	00:00:00	1654
1	02:00:00	881
2	04:00:00	658
3	06:00:00	1141
4	08:00:00	1581
5	10:00:00	1679
6	12:00:00	2266

Fig. Grouping Crimes Based on Equal Time Frames
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

2. Moving to bivariate analysis, we have tried to describe the relationship of number of crimes with the mean enrollment in schools and business establishments. The statistical information and visualizations derived from these will help us in showing the impact of crimes on businesses and schools.

Implementation

- Relationship between number of crimes and mean enrollment in schools: Using group by function we have grouped the number of crimes and mean enrollment in each year from 2020 to 2024 and visualized using double bar-chart to show how the crimes are impacting mean enrollment.

```
[127]: tb5= f.groupby(f['Year_x'])['Enrollment'].mean()
tb6= f.groupby(f['Year_x'])['DR_NO'].count()
tb5= tb5.reset_index()
tb6= tb6.reset_index()
pd1= pd.concat([tb5,tb6['DR_NO']],axis=1)
pd1=pd1.rename(columns={'Year_x':'Year','Enrollment':'Mean_Enrollment','DR_NO':'Number of Crimes'})
pd1
```

[127]:

	Year	Mean_Enrollment	Number of Crimes
0	2020	368.152107	4201
1	2021	375.553748	4549
2	2022	369.622822	4706
3	2023	371.879614	4768
4	2024	365.076439	2067

Fig. Grouping Crimes Based on Mean Enrollment in Schools in each Year
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

- Relationship between number of crimes and business establishments: Using group by function we have grouped the number of crimes and business establishments in each year from 2020 to 2024 and visualized using double bar-chart to show how the crimes are impacting business establishments.

[26]:
```
count_2020=0
count_2021=0
count_2022=0
count_2023=0
count_2024=0
for i in Year_Business.index:
    if Year_Business.loc[i,'Year']=='2020':
        count_2020=count_2020+1
    elif Year_Business.loc[i,'Year']=='2021':
        count_2021=count_2021+1
    elif Year_Business.loc[i,'Year']=='2022':
        count_2022=count_2022+1
    elif Year_Business.loc[i,'Year']=='2023':
        count_2023=count_2023+1
    elif Year_Business.loc[i,'Year']=='2024':
        count_2024=count_2024+1
l1=[count_2020,count_2021,count_2022,count_2023,count_2024]
l2=['2020','2021','2022','2023','2024']
pd1=pd.DataFrame(l1,columns=['Number of Businesses'])
pd2=pd.DataFrame(l2,columns=['Year'])
pd3=pd.concat([pd2,pd1,tb6['DR_NO']],axis=1)
pd3=pd3.rename(columns={'DR_NO':'Number of Crimes'})
```

[27]:
```
pd3
```

[27]:

	Year	Number of Businesses	Number of Crimes
0	2020	296	4201
1	2021	0	4549
2	2022	69	4706
3	2023	369	4768
4	2024	0	2067

Fig. Grouping Number of Crimes based on Number of Businesses in each Year
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

3. Finally for multivariate analysis, we have involved the number of schools and businesses with respect to crimes in a specific neighborhood. We have tried to categorize the neighborhood as safe or unsafe to defend our hypothesis.

Implementation

- Categorizing neighborhood into safe or unsafe: Initially we have categorized neighborhoods into its broader region i.e. Northern, Western, Central, Eastern and Southern. We then grouped the number of crimes based on the number of schools of each type and their respective enrollments. Similarly we grouped the number of crimes based on the type of businesses their respective count. Finally, we have merged the above two tables and we have visualized to show which neighborhood is safe.

```
[184]: f['Neighbourhood'].unique()

[184]: array(['South Los Angeles', 'Downtown Los Angeles', 'Koreatown',
       'Sherman Oaks', 'Northeast Los Angeles', 'Echo Park', 'Central LA',
       'Westlake', 'Westwood', nan, 'Pico-Union', 'Fairfax', 'Florence',
       'Mount Washington', 'Watts'], dtype=object)

[190…  region_mapping = {
           'South Los Angeles': 'Southern',
           'Downtown Los Angeles': 'Central',
           'Koreatown': 'Central',
           'Sherman Oaks': 'Western',
           'Northeast Los Angeles': 'Eastern',
           'Echo Park': 'Central',
           'Central LA': 'Central',
           'Westlake': 'Central',
           'Westwood': 'Western',
           'Pico-Union': 'Central',
           'Fairfax': 'Central',
           'Florence': 'Southern',
           'Mount Washington': 'Northern',
           'Watts': 'Southern'
       }

[191]: f['Categorised_Nhoods']=f['Neighbourhood'].map(region_mapping)

[192]: f['Categorised_Nhoods']

[192]: 0         Southern
       1         Southern
       2         Southern
       3         Southern
       4         Southern
                   ...
       20286     Central
       20287     Central
       20288     Central
       20289     Central
       20290     Central
       Name: Categorised_Nhoods, Length: 20291, dtype: object
```

Fig. Grouping Neighborhoods based on its broader regions
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

```
[214]: tb9=f.groupby(['Categorised_Nhoods','Category2'])['DR_NO'].count().sort_index()
       tb9=tb9.reset_index()
       tb10=pd.merge(tb9,tb8,on='Category2',how='left')
       tb10=tb10.rename(columns={'DR_NO':'Number of Schools','Category2':'TypeOfSchools'})

[215]: tb10
```

[215]:

	Categorised_Nhoods	TypeOfSchools	Number of Schools	Enrollment
0	Central	Elementary Schools	2866	373.684149
1	Central	Elementary-High Combination Schools	304	941.000000
2	Central	High Schools	794	235.271725
3	Central	Intermediate/Middle/Junior High Schools	422	407.345007
4	Eastern	High Schools	360	235.271725
5	Northern	Elementary Schools	13	373.684149
6	Southern	Elementary Schools	7700	373.684149
7	Southern	High Schools	1782	235.271725
8	Southern	Intermediate/Middle/Junior High Schools	5346	407.345007
9	Western	Elementary Schools	64	373.684149

Fig. Grouping Number of Schools and its Mean Enrollment based on Type of Schools in each of Categorised Neighborhoods
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

```
[224]: tb11=pd.merge(tb10,tb7,on='Categorised_Nhoods',how='left')
       tb11=tb11.rename(columns={'count':'Number of Crimes'})
       tb11
```

[224]:

	Categorised_Nhoods	TypeOfSchools	Number of Schools	Enrollment	Number of Crimes
0	Central	Elementary Schools	2866	373.684149	4386
1	Central	Elementary-High Combination Schools	304	941.000000	4386
2	Central	High Schools	794	235.271725	4386
3	Central	Intermediate/Middle/Junior High Schools	422	407.345007	4386
4	Eastern	High Schools	360	235.271725	360
5	Northern	Elementary Schools	13	373.684149	13
6	Southern	Elementary Schools	7700	373.684149	14828
7	Southern	High Schools	1782	235.271725	14828
8	Southern	Intermediate/Middle/Junior High Schools	5346	407.345007	14828
9	Western	Elementary Schools	64	373.684149	212
10	Western	High Schools	148	235.271725	212

Fig. Grouping Number of Crimes based on Number of Schools of each type and its Mean Enrollment in each of Categorised Neighborhoods
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

```
[244]: tb13=f.groupby(['Categorised_Nhoods','Categorised_Businesses'])['DR_NO'].count().sort_index()
       tb13=tb13.reset_index()
       tb13=tb13.rename(columns={'DR_NO':'Number of Businesses'})
       tb14=pd.merge(tb7,tb13,on='Categorised_Nhoods',how='right')
       tb14=tb14.rename(columns={'count':'Number of Crimes'})
```

```
[245]: tb14
```

[245]:

	Categorised_Nhoods	Number of Crimes	Categorised_Businesses	Number of Businesses
0	Central	4386	Arts & Culture	126
1	Central	4386	Business Services	1597
2	Central	4386	Community Services	863
3	Central	4386	Education	126
4	Central	4386	Individual Contributors	1674
5	Eastern	360	Business Services	180
6	Eastern	360	Education	180
7	Northern	13	Community Services	13

Fig. Grouping Number of Crimes based on Number of Businesses of each type in each of Categorised Neighborhoods
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

```
[250]: tb15=pd.merge(tb10,tb14,on='Categorised_Nhoods',how='inner')
```

```
[251]: tb15
```

[251]:

	Categorised_Nhoods	TypeOfSchools	Number of Schools	Enrollment	Number of Crimes	Categorised_Businesses	Number of Businesses
0	Central	Elementary Schools	2866	373.684149	4386	Arts & Culture	126
1	Central	Elementary Schools	2866	373.684149	4386	Business Services	1597
2	Central	Elementary Schools	2866	373.684149	4386	Community Services	863
3	Central	Elementary Schools	2866	373.684149	4386	Education	126
4	Central	Elementary Schools	2866	373.684149	4386	Individual Contributors	1674
5	Central	Elementary-High Combination Schools	304	941.000000	4386	Arts & Culture	126
6	Central	Elementary-High Combination Schools	304	941.000000	4386	Business Services	1597

Fig. Concatenated result of above two tables showing Number of Crimes, Number of Businesses, Number of Schools in each of Categorised Neighbourhood
Tool - Jupyter ; **Language** - Python ; **Library** - seaborn

VISUALIZATIONS

We have Used Tableau and Python's Matplotlib to show Visualisations for each of the above grouped data done in exploratory data analysis.

Visualization of Distribution of number of crimes w.r.t. age group:
The bar chart shows the distribution of crimes across different age groups of victims. The 26-40 age group has the highest number of crimes, followed by the 41-70 age group. The 0-10 age group has the lowest number of crimes.

Using Python:

```
tb1=f['Vict_AgeGrp'].value_counts().sort_index()
tb1=tb1.reset_index()
plt.bar(tb1['Vict_AgeGrp'],tb1['count'],color='green',edgecolor='black')
plt.xlabel('Age Group')
plt.ylabel('No of Crimes')
plt.title('No of Crimes in each Age Group')
plt.show()
```

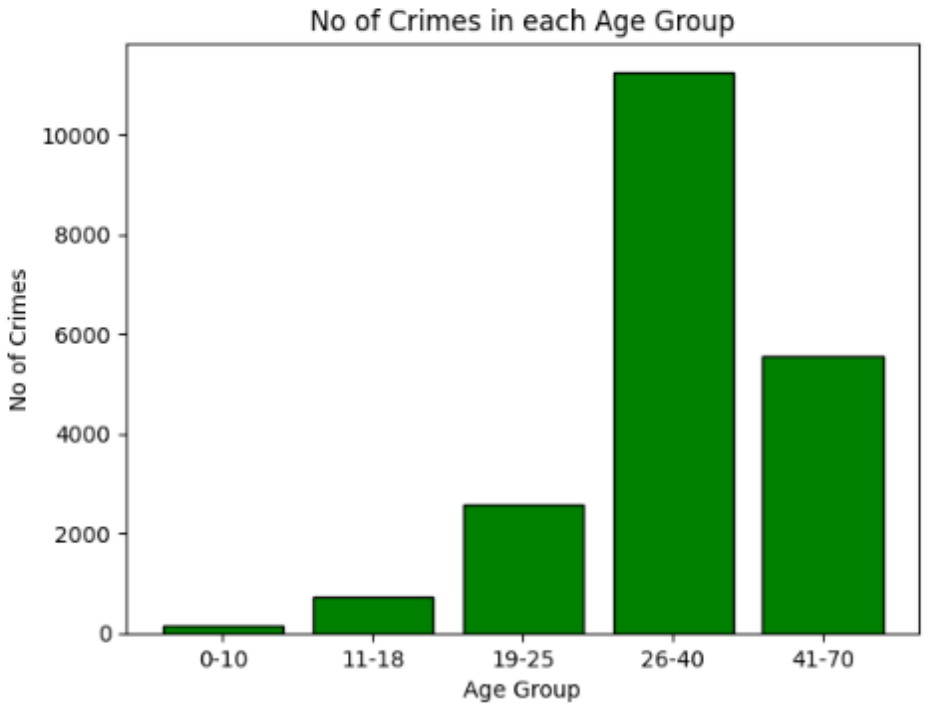

Fig. Distribution of number of crimes w.r.t. age group
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

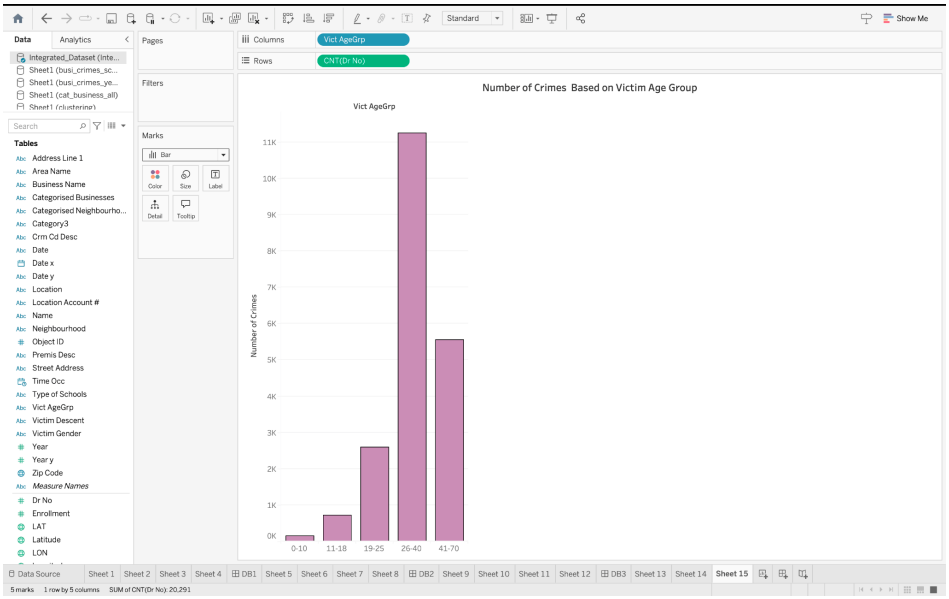

Fig. Distribution of number of crimes w.r.t. age group
Tool -

Visualization of Distribution of number of crimes w.r.t. gender:

The bar chart shows the distribution of crimes across different genders. The majority of crimes occurred with female victims, followed by male victims. A small number of crimes involved victims with an unknown gender.

Using Python:

```
tb2=f['Vict Sex'].value_counts().sort_index()
tb2=tb2.reset_index()
plt.bar(tb2['Vict Sex'],tb2['count'],color='red',edgecolor='black')
plt.xlabel('Gender')
plt.ylabel('No of Crimes')
plt.title('No of Crimes in each Gender')
plt.show()
```

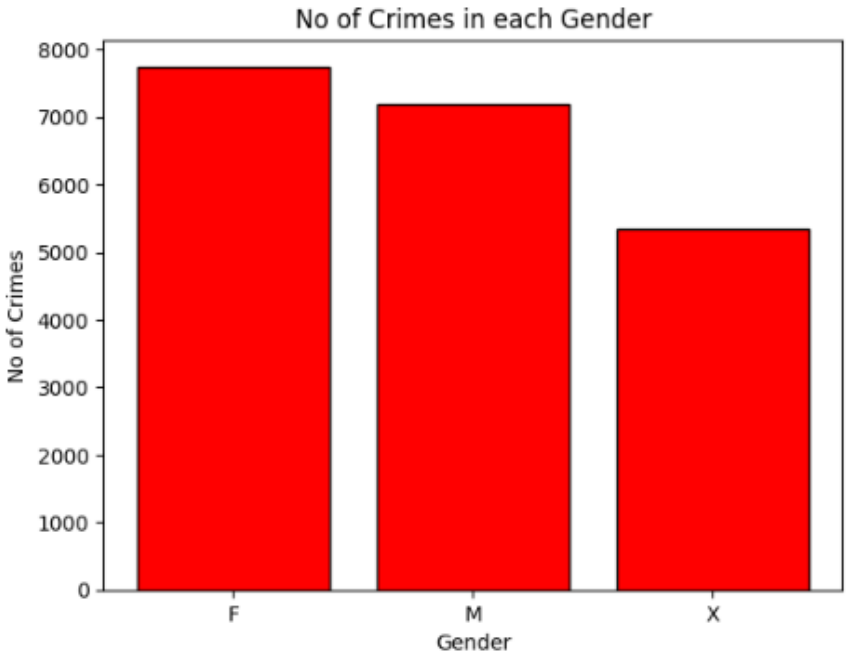

Fig. Distribution of number of crimes w.r.t. gender
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

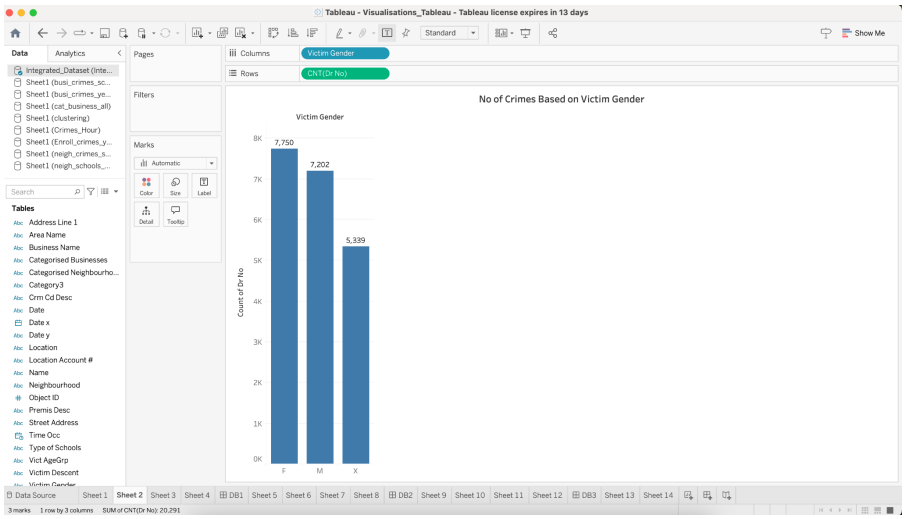

Fig. Distribution of number of crimes w.r.t. gender
Tool - Tableau

Visualization of Pie chart regarding the victim population in crimes w.r.t gender:

The pie chart shows the distribution of crimes across different genders. The majority of crimes occurred with female victims (38.2%), followed by male victims (35.5%). A smaller portion of crimes involved victims with an unknown gender (26.3%).

Using Python:

```
[18]: plt.pie(tb2['count'], labels=tb2['Vict Sex'], autopct='%1.1f%%', startangle=140)
      plt.title('Victim Population in crimes based on Gender')
```

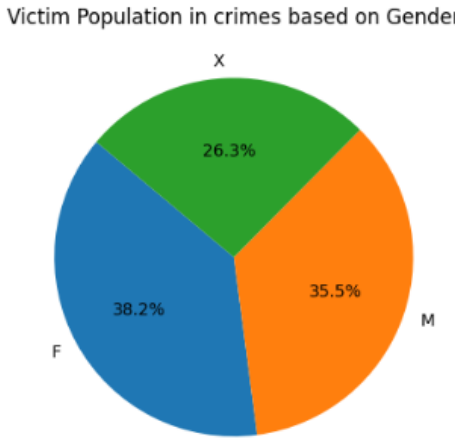

Fig. Pie chart regarding the victim population in crimes w.r.t gender
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

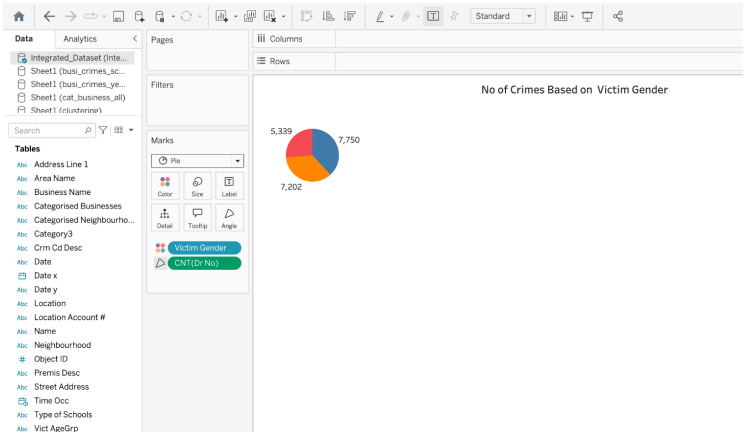

Fig. Pie chart regarding the victim population in crimes w.r.t gender
Tool - Tableau

Visualization of Distribution of crimes w.r.t. descent:

The bar chart shows the distribution of crimes across different descent groups. The descent group 'H' has the highest number of crimes, followed by 'X' and 'B'. Several descent groups have very few or no reported crimes.

Using Python:

```
9]: tb3=f['Vict Descent'].value_counts().sort_index()
    tb3=tb3.reset_index()
    plt.bar(tb3['Vict Descent'],tb3['count'],color='orange',edgecolor='ba)
    plt.xlabel('Descent')
    plt.ylabel('No of Crimes')
    plt.title('No of Crimes in each Descent')
    plt.show()
```

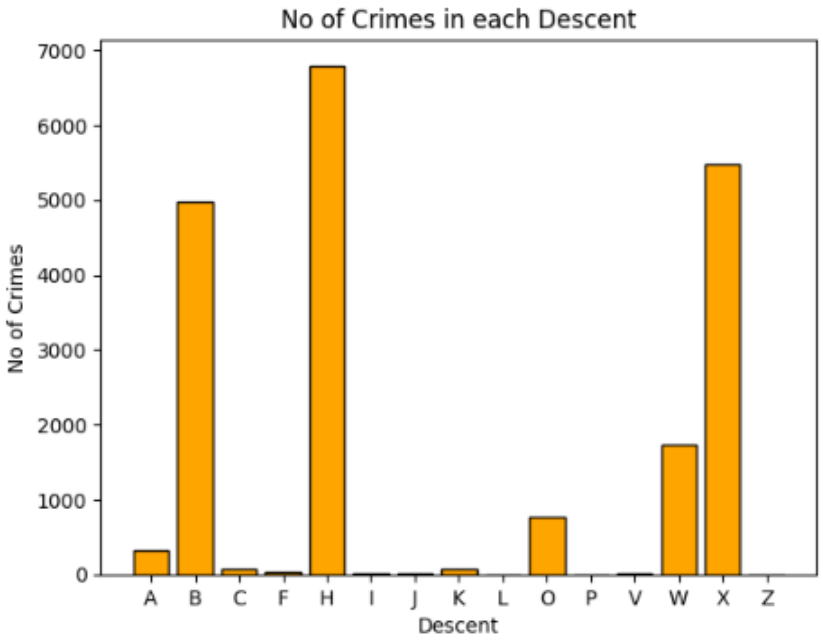

Fig. Distribution of crimes w.r.t. descent

Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

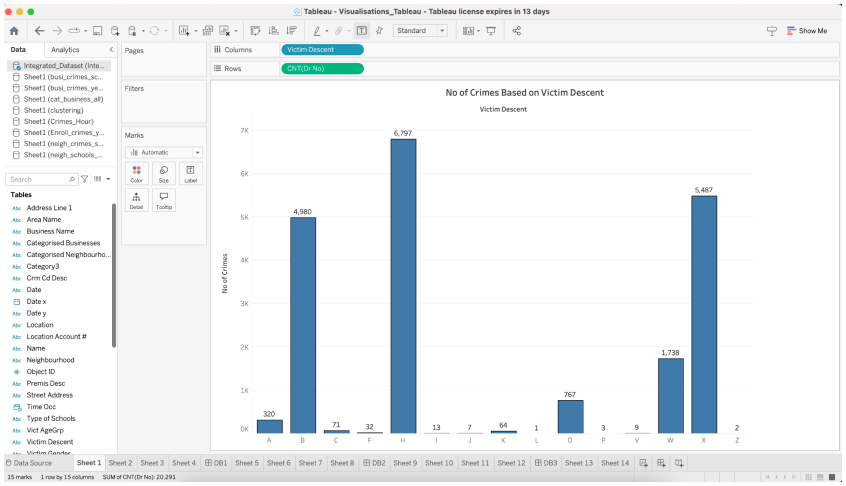

Fig. Distribution of crimes w.r.t. descent

Tool - Tableau

Visualization of Distribution of crimes w.r.t. time frame :

The bar chart shows the distribution of crimes across different time frames. The highest number of crimes occur between 14:00:00 and 16:00:00. The lowest number of crimes occur between 04:00:00 and 06:00:00.

Using Python:

```
[22]: plt.bar(tb4['Hour'],tb4['Count'],color='purple',edgecolor='black')
      plt.xlabel('Time Frame')
      plt.ylabel('No of Crimes')
      plt.title('No of Crimes in each Time Frame')
      plt.xticks(rotation=45)
      plt.show()
```

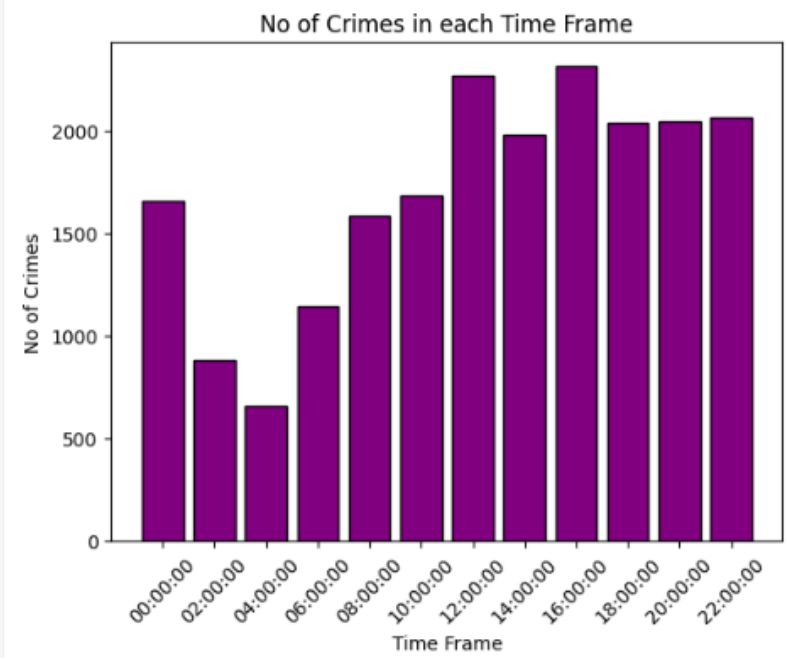

Fig. Distribution of crimes w.r.t. time frame
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

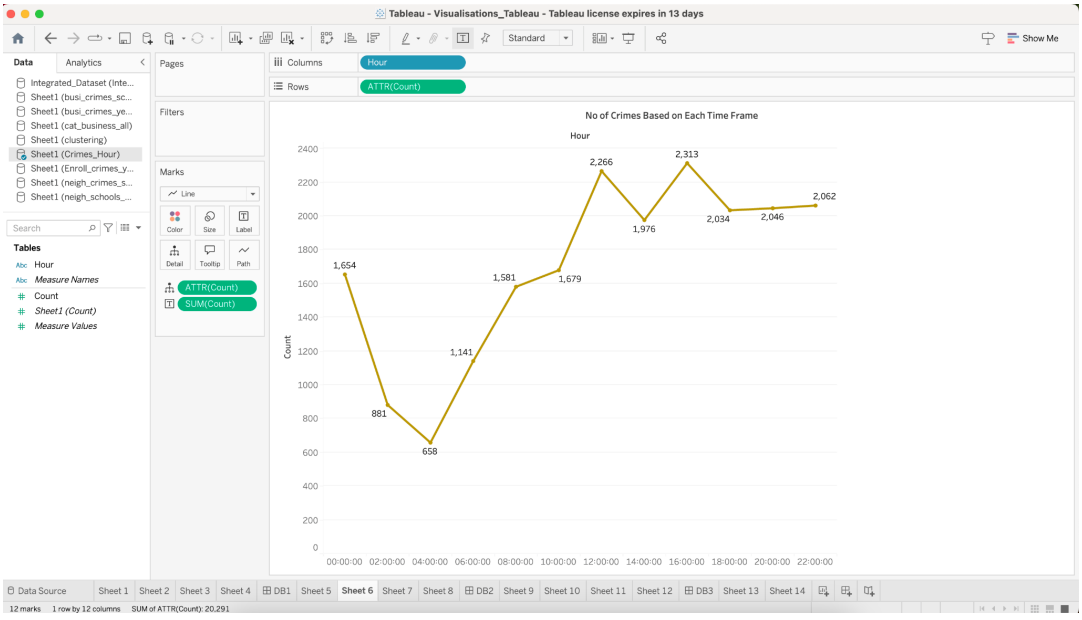

Fig. Distribution of crimes w.r.t. time frame
Tool - Tableau

Visualization of Impact of crimes on mean enrollment in schools :

The double bar chart shows the mean enrollment and number of crimes per year from 2020 to 2024. The mean enrollment and number of crimes are highest in 2022, followed by 2024 and 2023. The mean enrollment and number of crimes are lowest in 2020.

Using Python:

```
[24] Collapse Output .35
x = np.arange(5)
plt.bar(x - width/2, pd1['Mean_Enrollment'], width=width, label='Mean Enrollment', color='blue',)
plt.bar(x + width/2, pd1['Number of Crimes'], width=width, label='Number of Crimes', color='orange')
plt.xlabel('Year')
plt.ylabel('Values')
plt.xticks(x, pd1['Year'])
plt.legend()
plt.title('Mean_Enrollment and Number of Crimes Per Each Year')
for bar in bars1:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, str(yval),
             ha='center', va='bottom', color='black')
for bar in bars2:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, str(yval),
             ha='center', va='bottom', color='black')
plt.show()
```

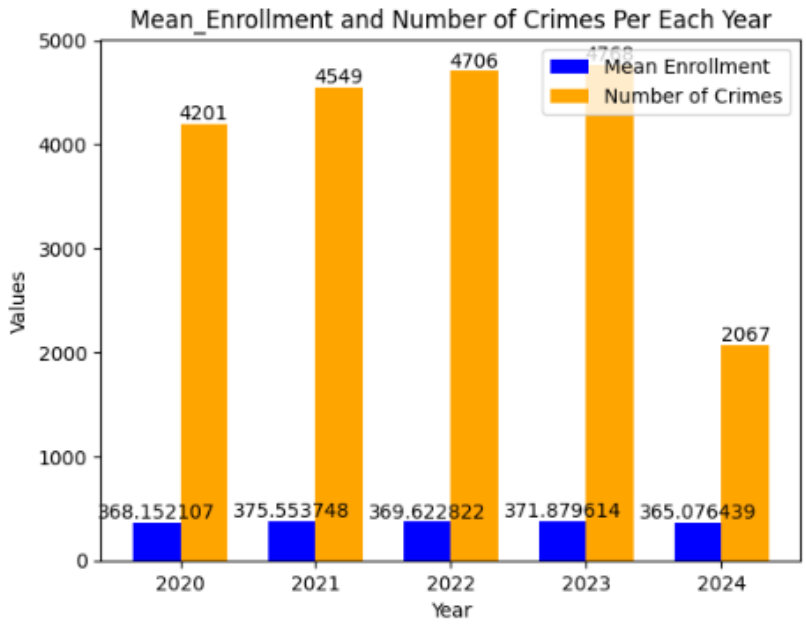

Fig. Impact of crimes on mean enrollment in schools

Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

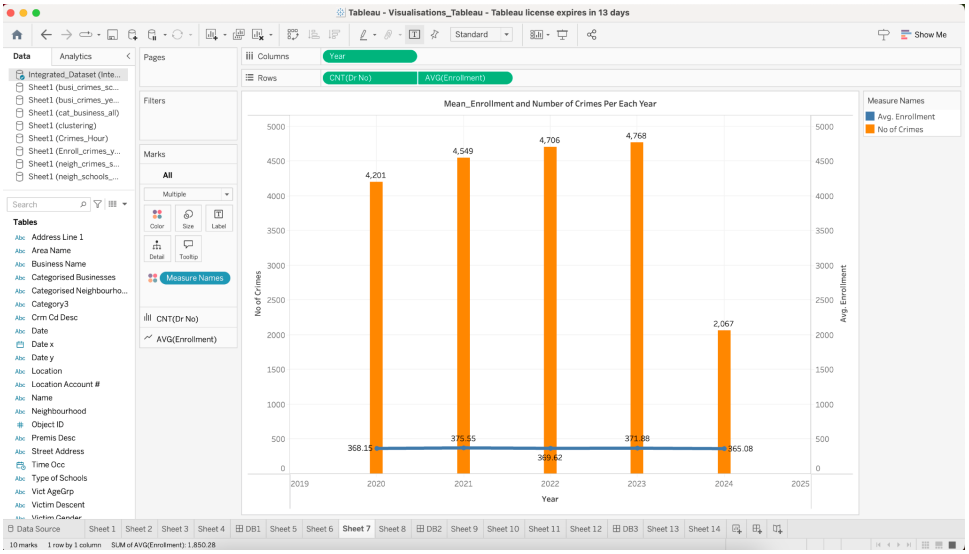

Fig. Impact of crimes on mean enrollment in schools
Tool - Tableau

Visualization of Impact of crimes on number of businesses:

The double bar chart shows the number of businesses and the number of crimes per year from 2020 to 2024. The number of businesses and crimes are highest in 2022, followed by 2024 and 2023. The number of businesses and crimes are lowest in 2020.

Using Python:

[181]:
```
width = 0.35
x = np.arange(5)
plt.bar(x - width/2, pd3['Number of Businesses'], width=width, label='Number of Businesses', color='blue',)
plt.bar(x + width/2, pd3['Number of Crimes'], width=width, label='Number of Crimes', color='orange')
plt.xlabel('Year')
plt.ylabel('Values')
plt.xticks(x, pd3['Year'])
plt.legend()
plt.title('Number of Businesses and Number of Crimes Per Each Year')
for bar in bars1:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, str(yval),
             ha='center', va='bottom', color='black')
for bar in bars2:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, str(yval),
             ha='center', va='bottom', color='black')
plt.show()
```

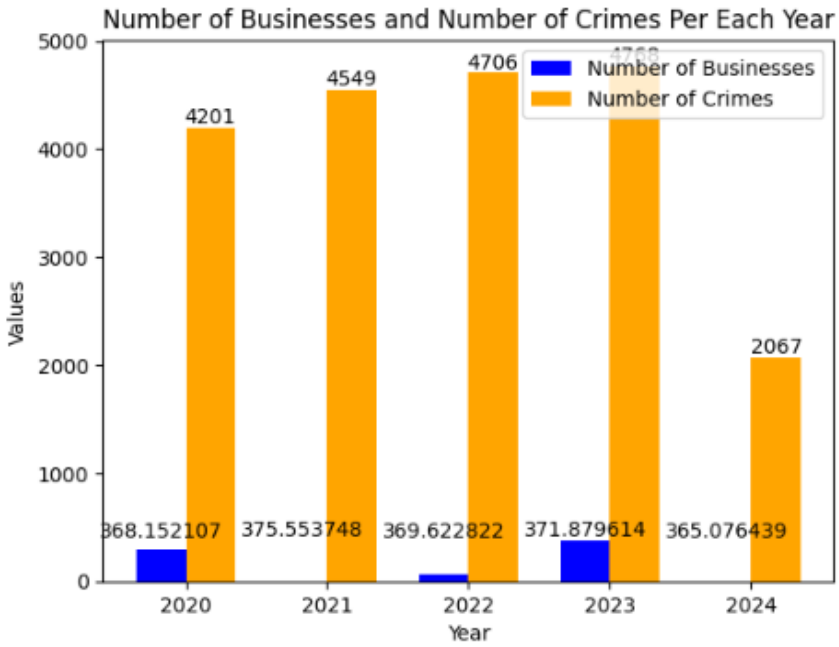

Fig. Impact of crimes on number of businesses
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

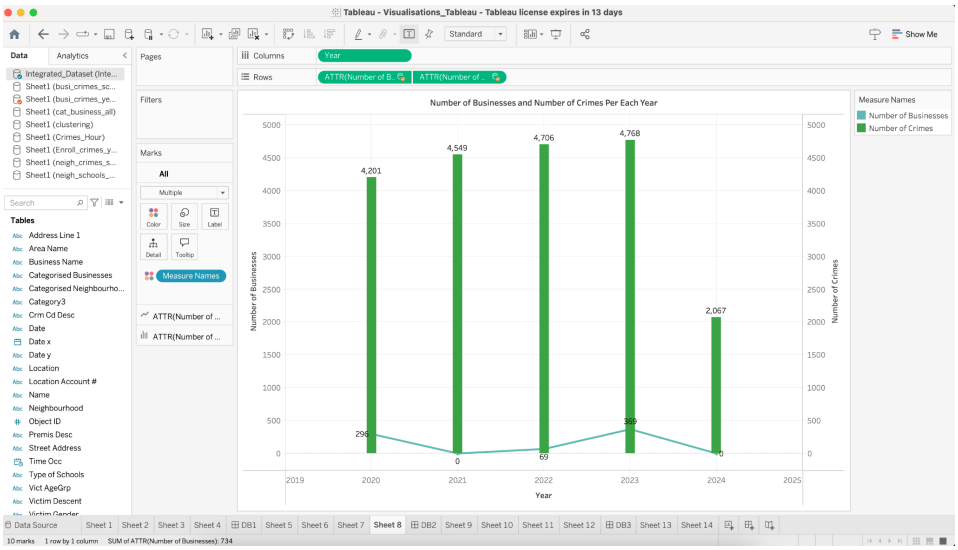

Fig. Impact of crimes on number of businesses
Tool - Tableau

Visualization of Distribution of Categorized neighborhoods:

The bar chart shows the distribution of crime across different regions. The Southern region has the highest number of crimes, followed by the Central region. The Eastern, Northern, and Western regions have significantly fewer crimes compared to the Southern and Central regions.

Using Python:

```
193]: tb7=f['Categorised_Nhoods'].value_counts().sort_index()
      tb7=tb7.reset_index()
      plt.bar(tb7['Categorised_Nhoods'],tb7['count'],color='pink',edgecolor='black')
```

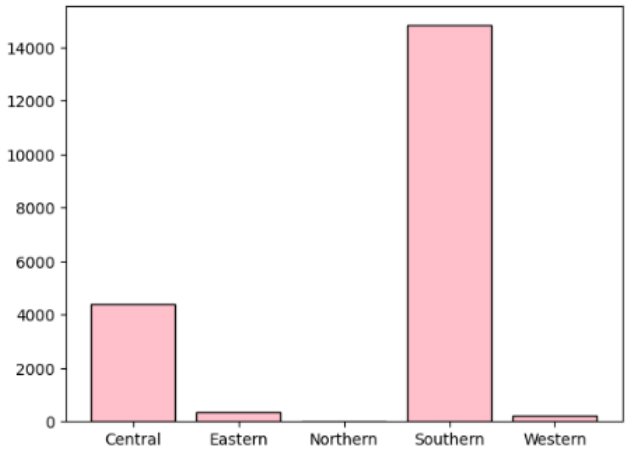

Fig. Distribution of Categorized neighborhoods
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

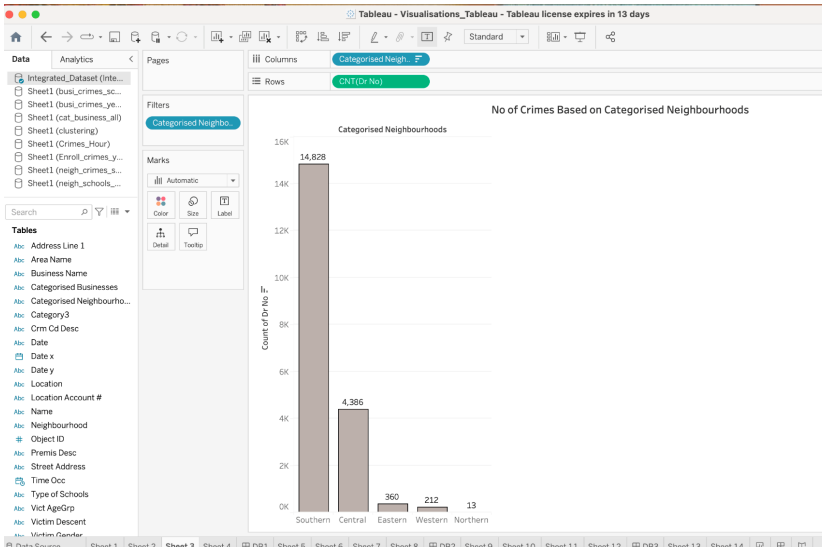

Fig. Distribution of Categorized neighborhoods
Tool - Tableau

Visualization of Distribution of enrollment w.r.t schools:

The bar chart shows the distribution of schools by type. Elementary-High Combination Schools have the highest number, followed by Elementary Schools and Intermediate/Middle/Junior High Schools. High Schools have the lowest number.

Using Python:

```
216]: tb8=f.groupby('Category2')['Enrollment'].mean().sort_index()
tb8=tb8.reset_index()
plt.bar(tb8['Category2'],tb8['Enrollment'],color='pink',edgecolor='black')
plt.xticks(rotation=45)
plt.show()
```

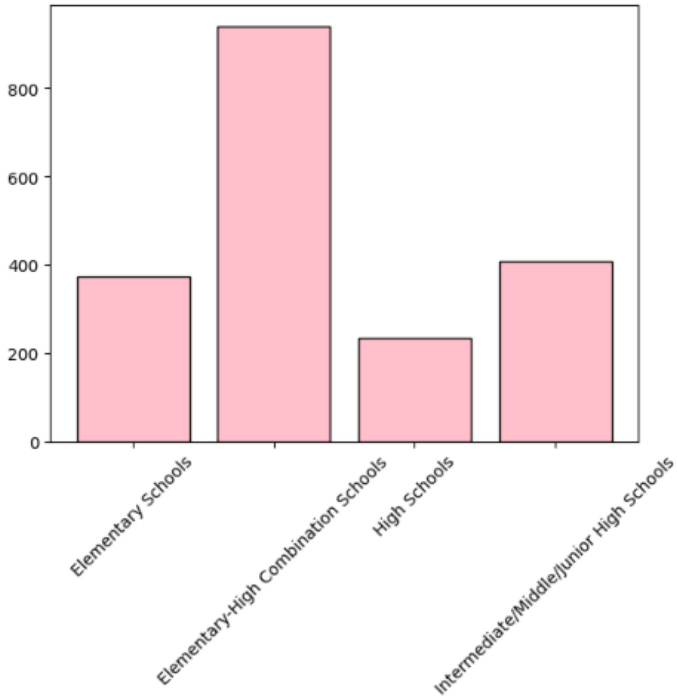

Fig. Distribution of enrollment w.r.t. schools
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

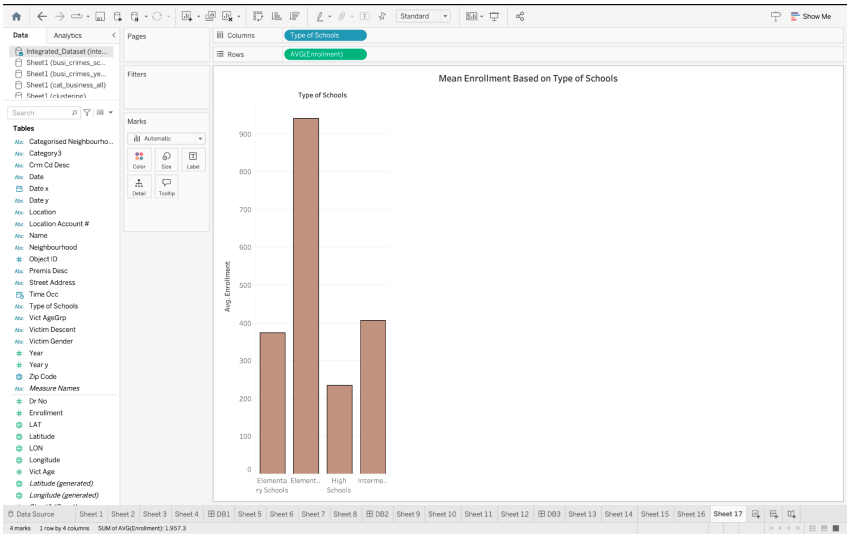

Fig. Distribution of enrollment w.r.t. schools
Tool - Tableau

Visualization of Scatter plot for number of schools w.r.t enrollment:

The scatter plot shows the relationship between the number of schools and enrollment in different neighborhoods. The Southern and Western neighborhoods have a higher number of schools and higher enrollment compared to the other neighborhoods. The Central and Eastern neighborhoods have a lower number of schools and lower enrollment.

Using Python:

```
[222]: plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=tb10,
                          x='Number of Schools',
                          y='Enrollment',
                          hue='Categorised_Nhoods',
                          legend=True)
plt.title('Scatter Plot of Number of Schools vs. Enrollment')
plt.xlabel('Number of Schools')
plt.ylabel('Enrollment')
plt.show()
```

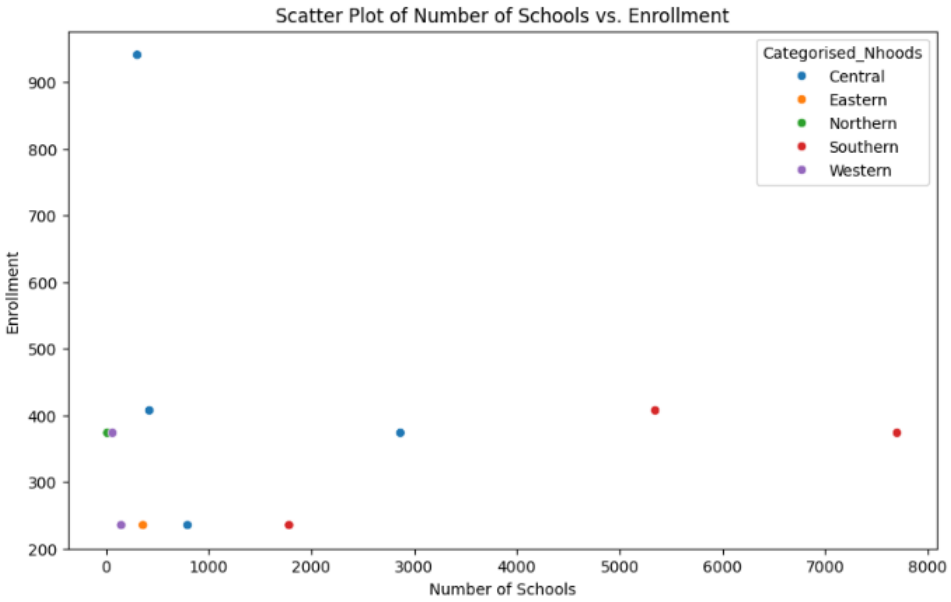

Fig. Scatter plot for number of schools w.r.t. enrollment
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

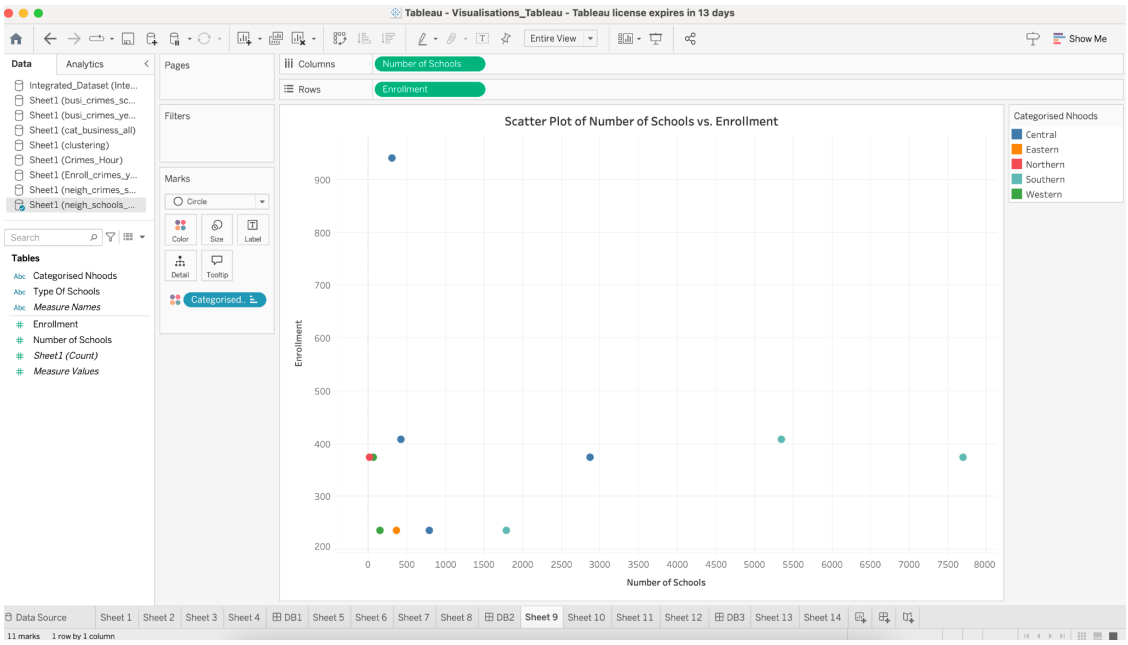

Fig. Scatter plot for number of schools w.r.t. enrollment
Tool - Tableau

Visualization of Scatter plot for number of schools w.r.t. number of crimes:
The scatter plot shows the relationship between the number of schools and the number of crimes in different neighborhoods.The Southern neighborhood has the highest number of crimes and a moderate number of schools. The Central and Eastern neighborhoods have a lower number of crimes and a lower number of schools.

Using Python:

```
[225]: plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=tb11,
                          x='Number of Schools',
                          y='Number of Crimes',
                          hue='Categorised_Nhoods',
                          legend=True)
plt.title('Scatter Plot of Number of Schools vs. Number of Crimes')
plt.xlabel('Number of Schools')
plt.ylabel('Number of Crimes')
plt.show()
```

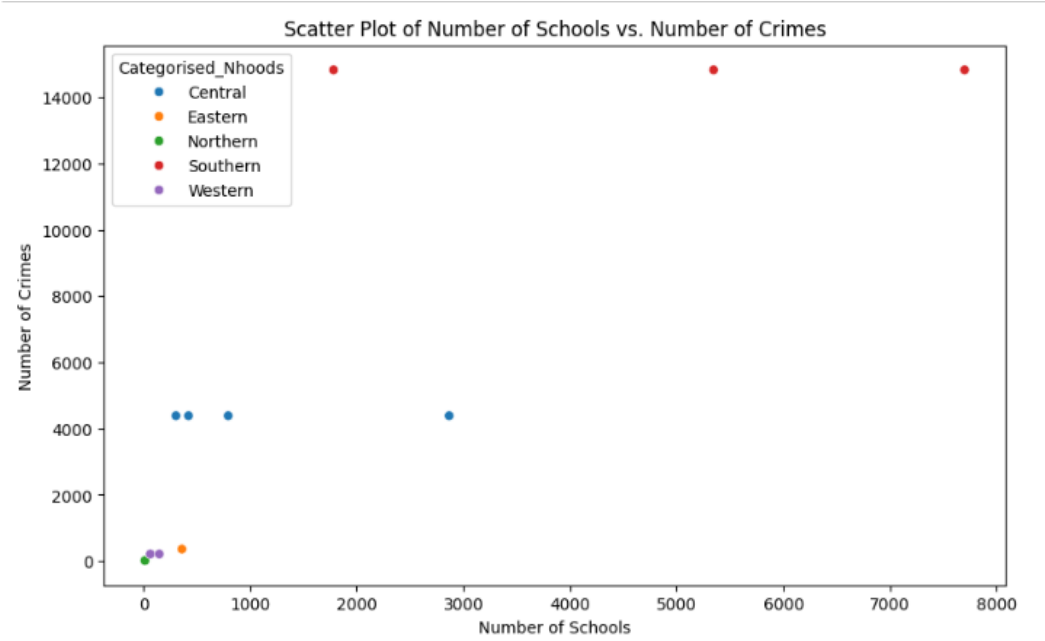

Fig. Scatter plot for number of schools w.r.t. number of crimes
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

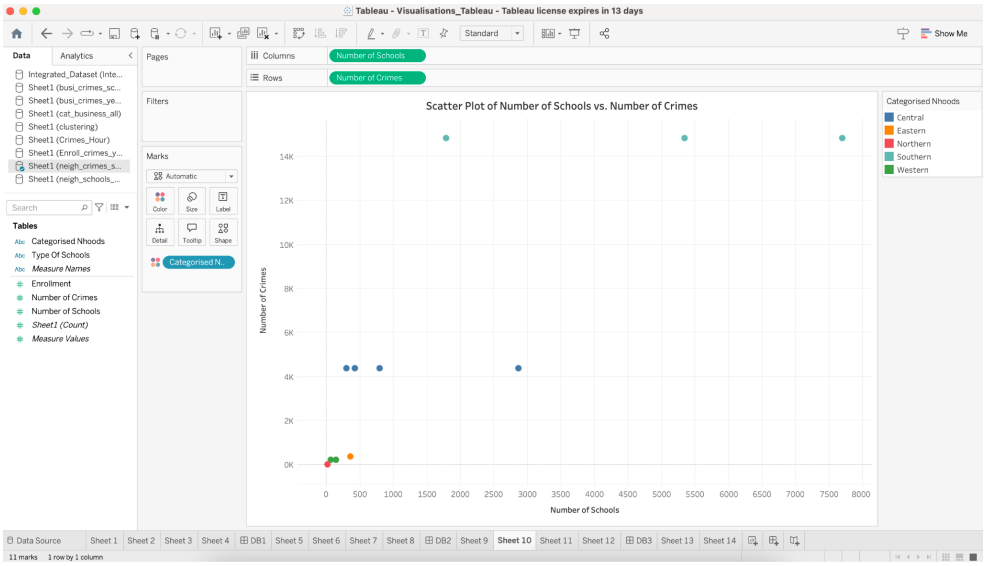

Fig. Scatter plot for number of schools w.r.t. number of crimes
Tool - Tableau

Visualization of Distribution of Categorized Businesses:

The bar chart shows the distribution of volunteers across different categories. The Education category has the highest number of volunteers, followed by Arts & Culture. Business Services, Community Services, and Individual Contributors have significantly fewer volunteers compared to the other categories.

Using Python:

```
[236]: tb12=f['Categorised_Businesses'].value_counts().sort_index()
       tb12=tb12.reset_index()
       plt.bar(tb12['Categorised_Businesses'],tb7['count'],edgecolor='black')
       plt.xticks(rotation=45)
       plt.show()
```

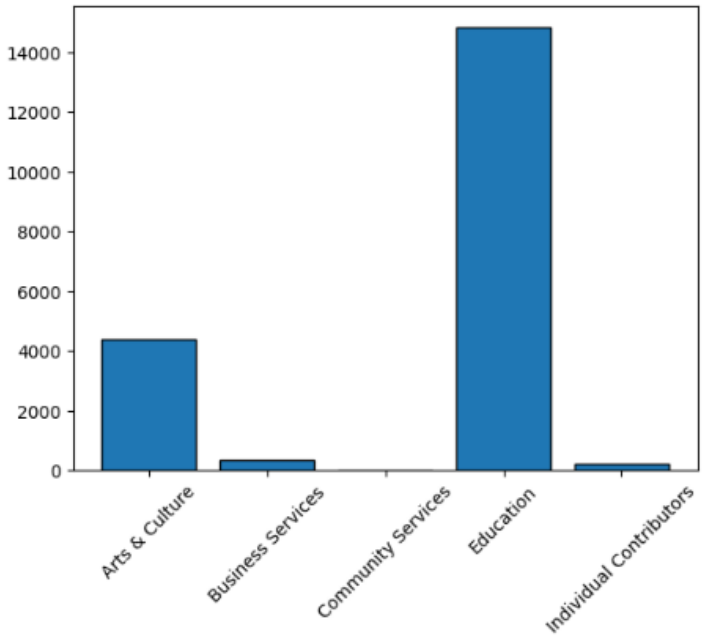

Fig. Distribution of Categorized Businesses
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

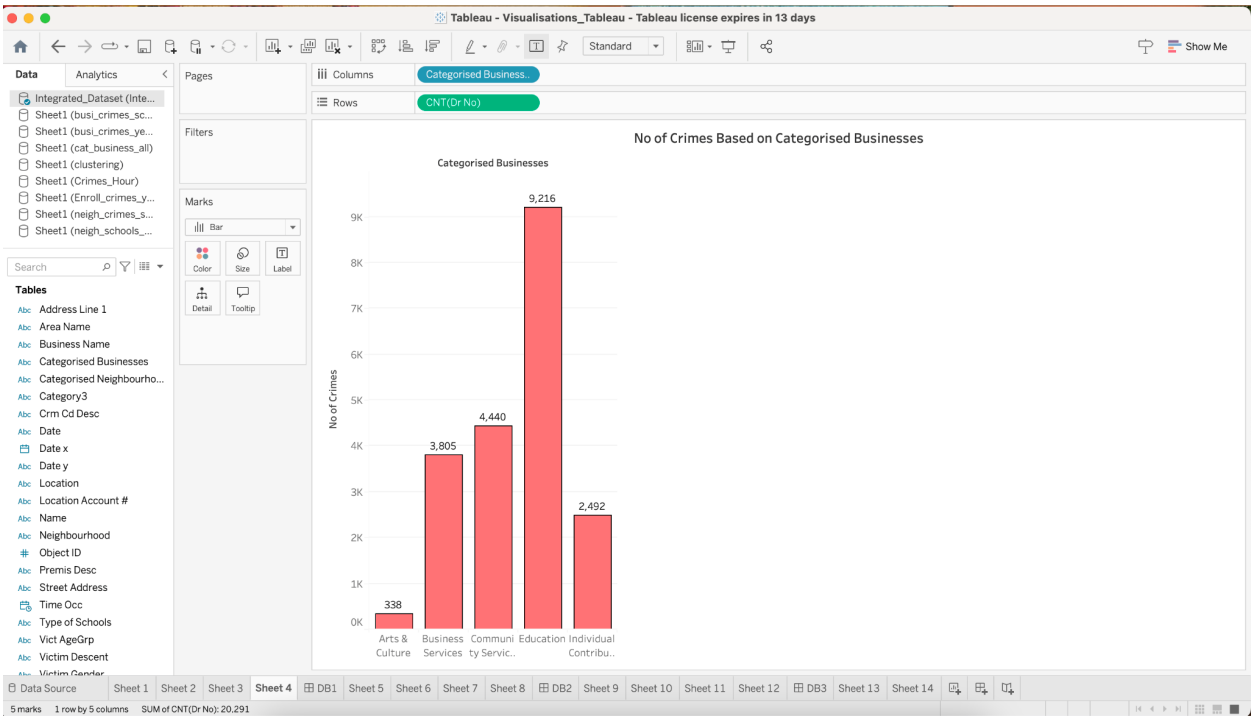

Fig. Distribution of Categorized Businesses
Tool - Tableau

Visualization of Scatter plot for categorized neighborhood w.r.t number of crimes :

The scatter plot shows the relationship between the number of businesses and the number of crimes in different neighborhoods. The Southern neighborhood has the highest number of crimes and a moderate number of businesses. The Central and Eastern neighborhoods have a lower number of crimes and a lower number of businesses.

Using Python:

[246]:
```
plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=tb14,
                          x='Number of Businesses',
                          y='Number of Crimes',
                          hue='Categorised_Nhoods',
                          legend=True)
plt.title('Scatter Plot of Number of Businesses vs. Number of Crimes')
plt.xlabel('Number of Businesses')
plt.ylabel('Number of Crimes')
plt.show()
```

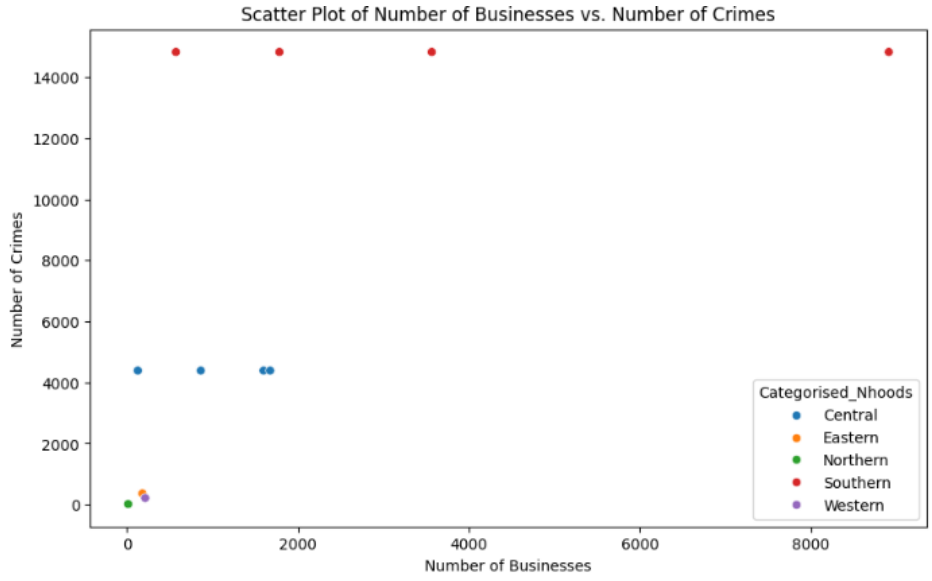

Fig. Scatter plot for categorized neighborhood w.r.t. number of crimes
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

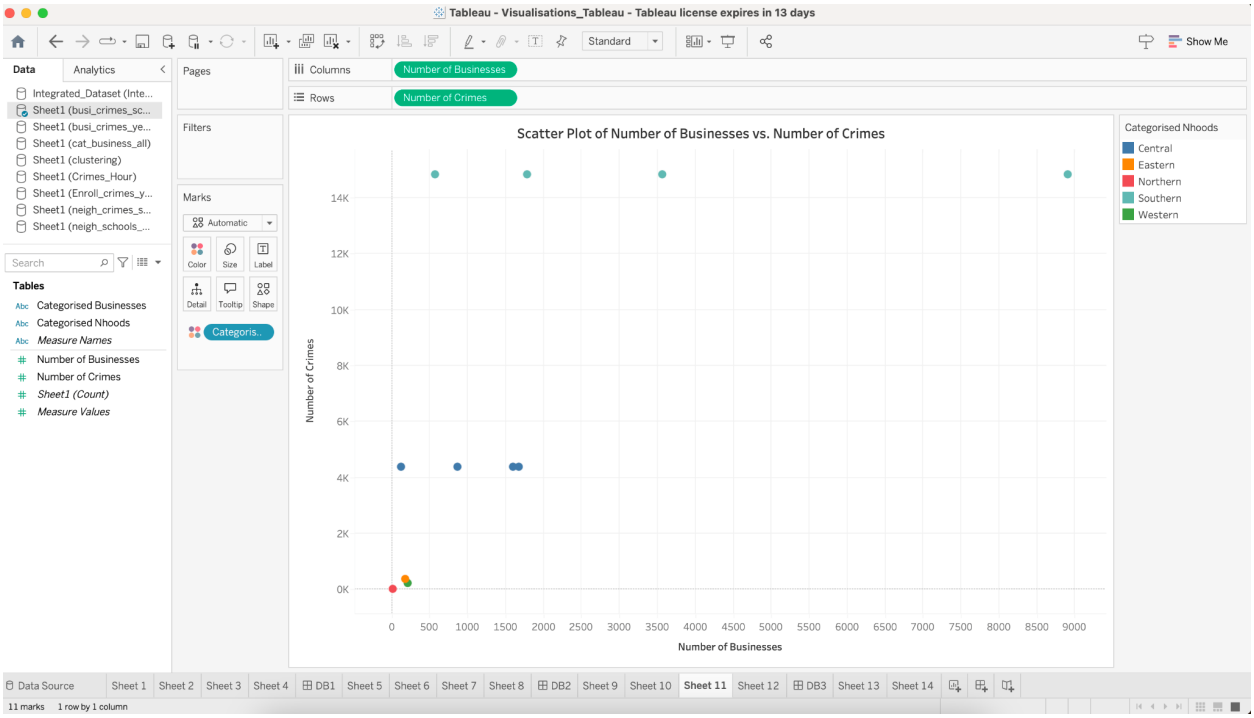

Fig. Scatter plot for categorized neighborhood w.r.t. number of crimes
Tool - Tableau

Visualization of Scatter plot for number businesses w.r.t number of crimes:

The scatter plot shows the relationship between the number of businesses and the number of crimes in different categories. The Education category has the highest number of crimes and a moderate number of businesses. The Arts & Culture and Business Services categories have a lower number of crimes and a lower number of businesses.

Using Python:

```
[247]: plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=tb14,
                          x='Number of Businesses',
                          y='Number of Crimes',
                          hue='Categorised_Businesses',
                          legend=True)
plt.title('Scatter Plot of Number of Businesses vs. Number of Crimes')
plt.xlabel('Number of Businesses')
plt.ylabel('Number of Crimes')
plt.show()
```

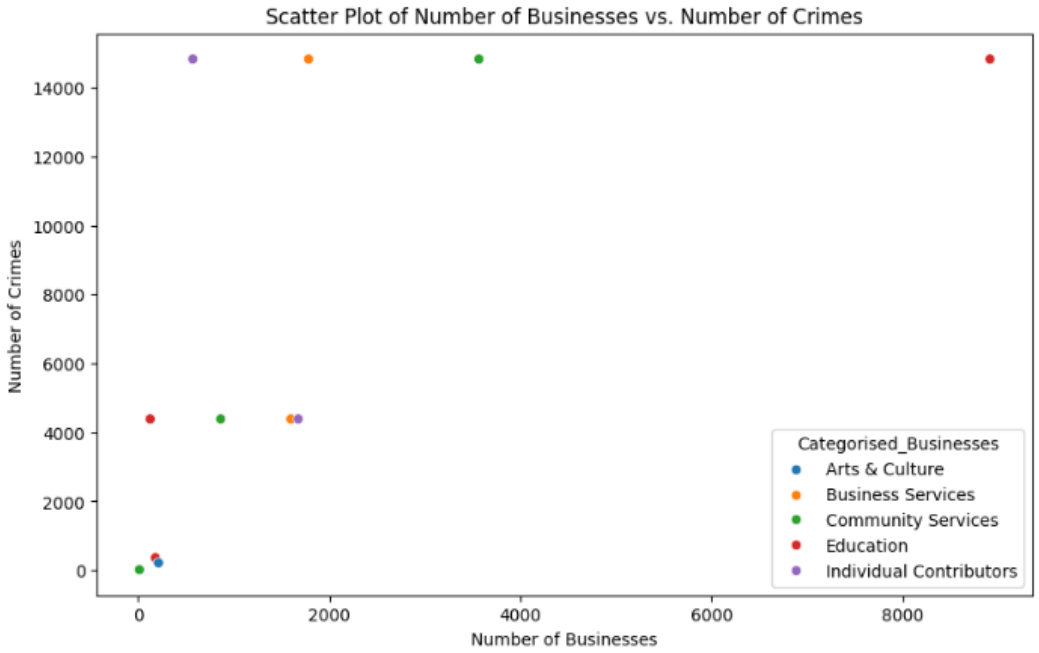

Fig. Scatter plot for number of businesses w.r.t. number of crimes
Tool - Jupyter ; **Language** - Python ; **Library** - matplotlib

Using Tableau:

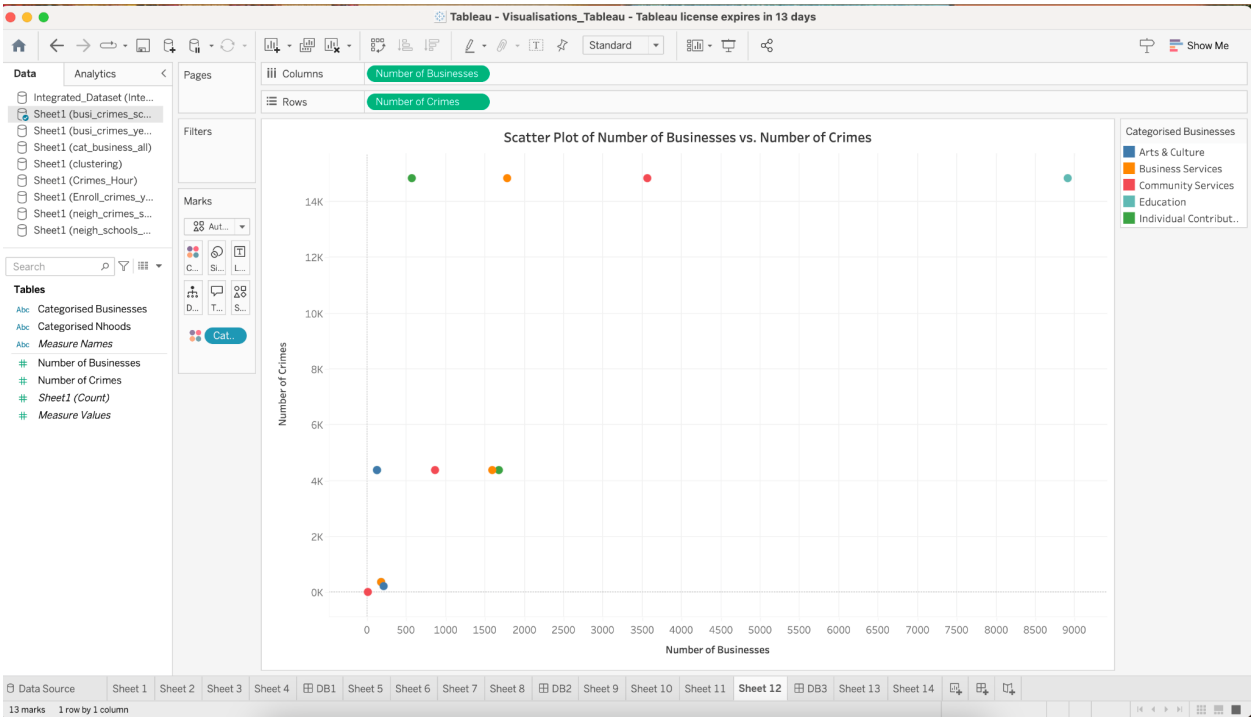

Fig. Scatter plot for number of businesses w.r.t. number of crimes
Tool - Tableau

Visualization of Stacked bar plot of categorized neighborhood w.r.t. schools and businesses:

The stacked bar chart shows the distribution of volunteers across different categories in each neighborhood.The Southern neighborhood has the highest number of volunteers in all categories, followed by the Central neighborhood. The Eastern, Northern, and Western neighborhoods have significantly fewer volunteers compared to the Southern and Central neighborhoods.The Education category has the highest number of volunteers in all neighborhoods, followed by Community Services and Arts & Culture. Business Services and Individual Contributors have the lowest number of volunteers in all neighborhoods.

Using Python:

```
x = np.arange(len(tb15['Categorised_Nhoods']))
width = 0.25
fig, ax = plt.subplots(figsize=(12, 6))

bars1 = ax.bar(x - width, tb15['Number of Crimes'], width, label='Number of Crimes', color='blue')
bars2 = ax.bar(x, tb15['Enrollment'], width, label='Mean Enrollment', color='green')
bars3 = ax.bar(x + width, tb15['Number of Businesses'], width, label='Number of Businesses', color='orange')

ax.set_xlabel('Categorised Neighborhoods')
ax.set_ylabel('Values')
ax.set_title('Number of Crimes,Mean Enrollment of Schools, Number of Businesses by Categorised Neighborhoods')
ax.set_xticks(x)
ax.set_xticklabels(tb15['Categorised_Nhoods'], rotation=45)
ax.legend()

for bars in [bars1, bars2, bars3]:
    ax.bar_label(bars, padding=3)

plt.tight_layout()
plt.show()
```

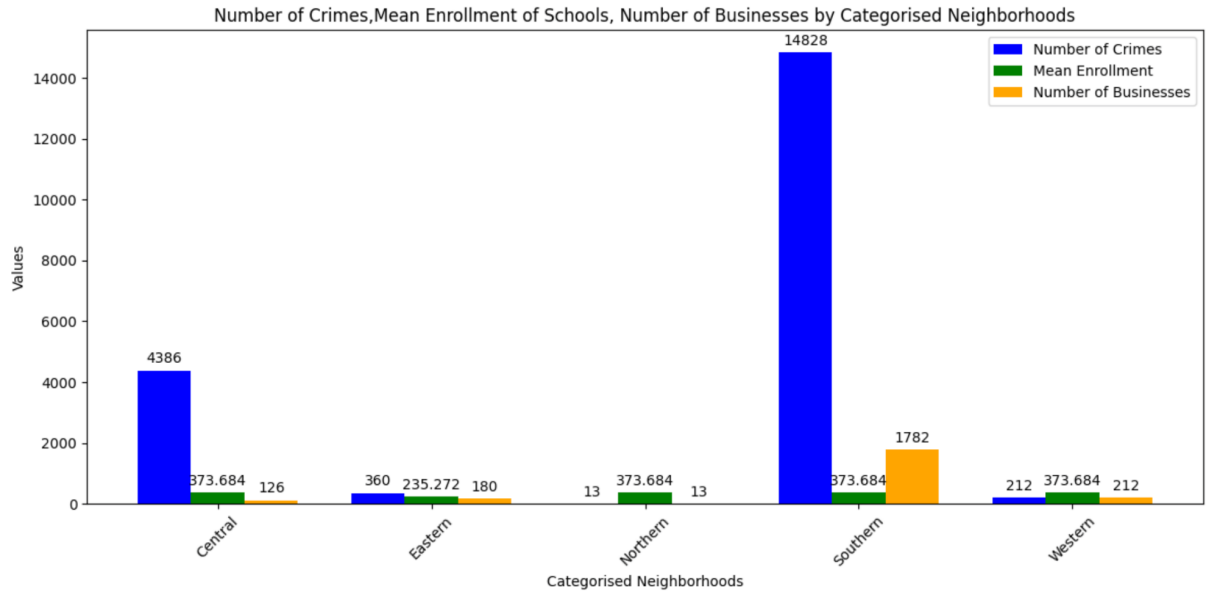

Fig. Stacked bar plot of categorized neighborhood w.r.t. schools and businesses

Tool - Jupyter ; **Language** - Python ; **Library** - seaborn

Using Tableau:

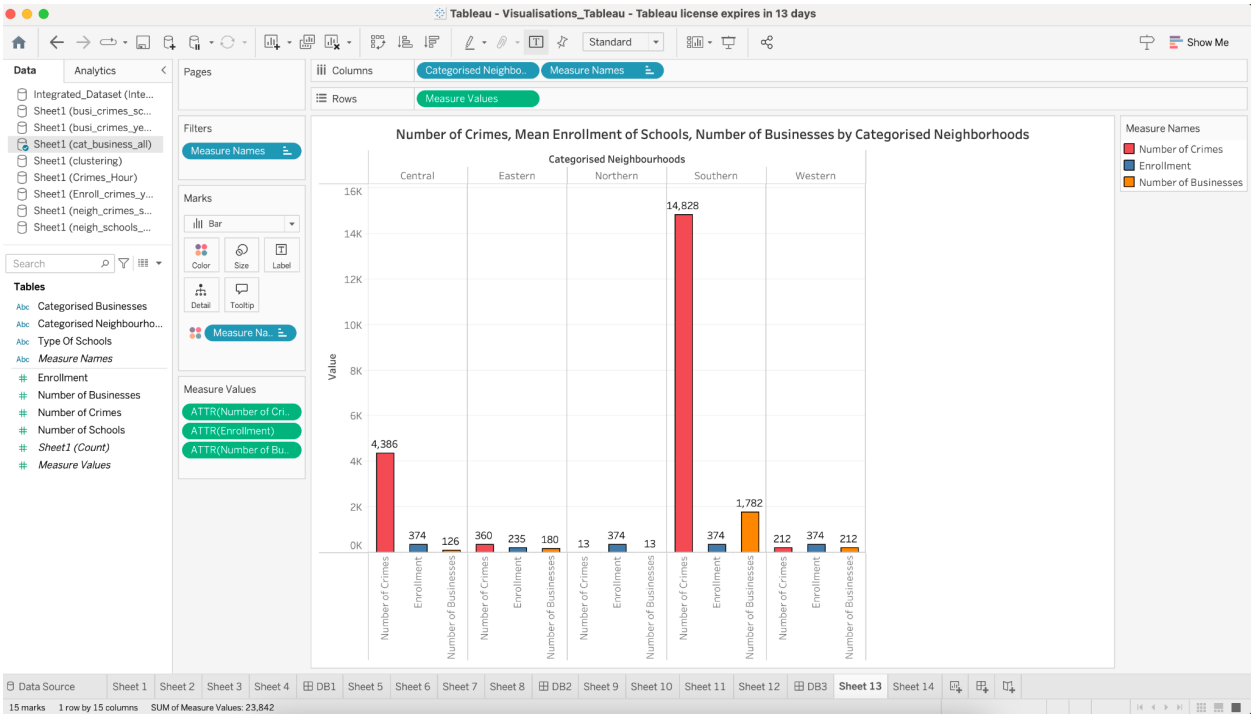

Fig. Stacked bar plot of categorized neighborhood w.r.t. schools and businesses
Tool - Tableau

INFORMATION MODELING

Algorithms Employed:

1. K-means Clustering
2. Decision Tree Classifier

After End Step of Exploratory Data Analysis where we got summary of Number of Crimes, Number of Business and Mean Enrollment in each of Categorised Neighbourhood, our goal is to build a model which takes the summary as input, train and categorise the data Neighbourhoods into high crime, low crime and medium crime.

Steps for Implementation

1. Importing Libraries

```
[44]: import pandas as pd
      from sklearn.cluster import KMeans
      import matplotlib.pyplot as plt
      from matplotlib.patches import Patch
      from sklearn.tree import DecisionTreeClassifier, plot_tree
      from sklearn.model_selection import train_test_split
```

Fig. Importing Libraries
Tool - Jupyter ; **Language** - Python

2. Importing the Summary Dataset

```
[2]: f=pd.read_excel('cat_business_all.xlsx')
```

```
[3]: f
```

[3]:

	Categorised_Nhoods	TypeOfSchools	Number of Schools	Enrollment	Number of Crimes	Categorised_Businesses	Number of Businesses
0	Central	Elementary Schools	2866	373.684149	4386	Arts & Culture	126
1	Eastern	High Schools	360	235.271725	360	Business Services	180
2	Northern	Elementary Schools	13	373.684149	13	Community Services	13
3	Southern	Elementary Schools	7700	373.684149	14828	Business Services	1782
4	Western	Elementary Schools	64	373.684149	212	Arts & Culture	212

Fig. Importing the Summary Dataset
Tool - Jupyter ; **Language** - Python

3. Data Normalisation:

   As Values of Independent Variables ie. Number Schools, Businesses, Crimes and Enrollment are of the same scale, there is no requirement of normalisation and model's performance wont get affected because of that.

4. Defining Model K-means, Training and Prediction

```
[28]: kmeans = KMeans(n_clusters=3, random_state=42)
```

```
[29]: f["Cluster"] = kmeans.fit_predict(f[["Number of Crimes", "Number of Businesses", "Enrollment"]])
```

Fig. Defining Model K-means, Training and Prediction
Tool - Jupyter ; **Language** - Python

5. Visualising the Clusters

Using Python:

```
[36]: colors = {0: 'red', 1: 'blue', 2: 'green'}
plt.figure(figsize=(10, 6))
for cluster in f["Cluster"].unique():
    cluster_data = f[f["Cluster"] == cluster]
    plt.scatter(
        cluster_data["Number of Businesses"],
        cluster_data["Number of Crimes"],
        label=f"Cluster {cluster}",
        color=colors[cluster],
        s=cluster_data["Enrollment"] * 10,
        alpha=0.7
    )
    for _, row in cluster_data.iterrows():
        plt.text(row["Number of Businesses"], row["Number of Crimes"], row["Categorised_Nhoods"], fontsize=9)
legend_handles = [Patch(color=colors[i], label=f"Cluster {i}") for i in colors]
plt.legend(handles=legend_handles, loc='lower right')
plt.title("Neighborhood Clustering (Crimes, Businesses, Enrollment)")
plt.xlabel("Number of Businesses")
plt.ylabel("Number of Crimes")
plt.grid(True)
plt.show()
```

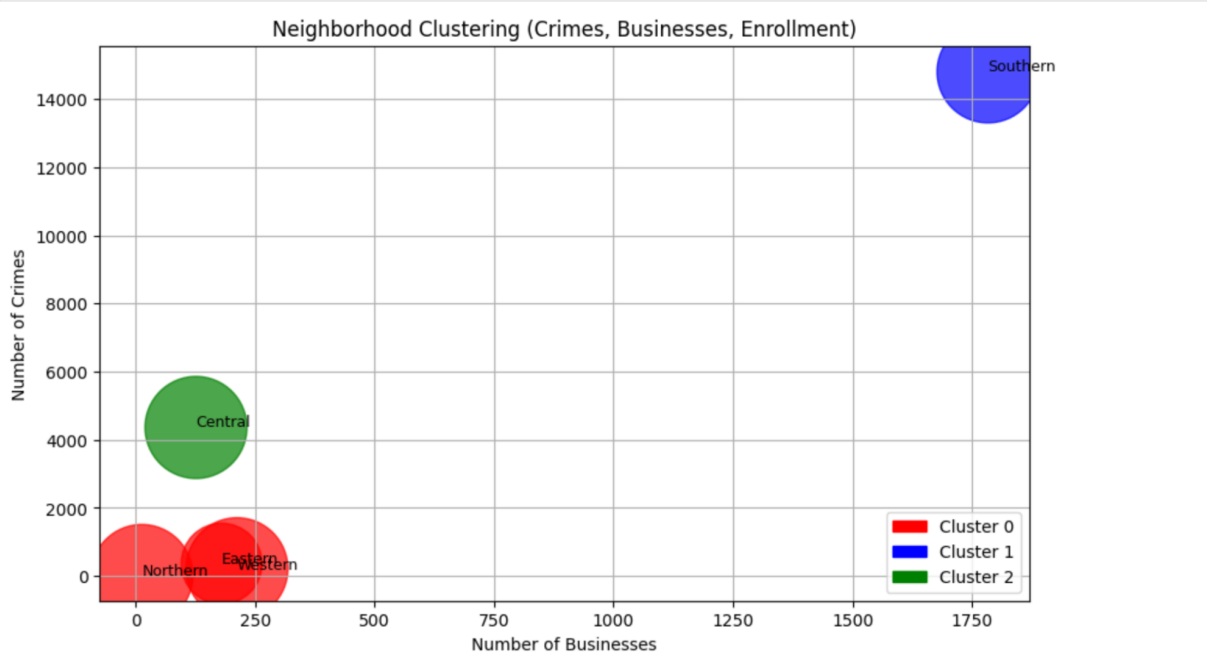

Fig. Visualising the Clusters
Tool - Jupyter ; **Language** - Python

Using Tableau:

Fig. Visualising the Clusters
Tool - Tableau

6. Implementing Decision Tree Classifier: As it is a supervised algorithm we have taken all the summary data for training, for testing purposes we have taken raw data similarly so that we evaluate the model performance.

Number of Businesses

```
[80]: x=f[['Enrollment','Number of Crimes','Number of Businesses']]

[81]: y=f['Categorised_Nhoods']

[82]: Model=DecisionTreeClassifier()
      Model.fit(x,y)

[82]: DecisionTreeClassifier
      DecisionTreeClassifier()
```

```
[87]: test_data = {
          "Categorised_Nhoods": ["Central", "Eastern", "Northern", "Western", "Central"],
          "Enrollment": [350.25, 300.75, 400.10, 375.50, 350.00],
          "Number of Crimes": [4500, 1200, 50, 15000, 300],
          "Number of Businesses": [150, 100, 10, 2000, 80]
      }
      test_data=pd.DataFrame(test_data)

[91]: xtest=test_data[['Enrollment','Number of Crimes','Number of Businesses']]
      ytest=test_data['Categorised_Nhoods']
      pred=Model.predict(xtest)

[95]: print(classification_report(ytest, pred))
```

```
                precision    recall  f1-score   support

       Central       1.00      1.00      1.00         2
       Eastern       1.00      1.00      1.00         1
      Northern       1.00      1.00      1.00         1
      Southern       0.00      0.00      0.00         0
       Western       0.00      0.00      0.00         1

      accuracy                           0.80         5
     macro avg       0.60      0.60      0.60         5
  weighted avg       0.80      0.80      0.80         5
```

Fig. Implementing Decision Tree Classifier
Tool - Jupyter ; **Language** - Python

EFFECTIVENESS OF THE ANALYSIS

Our analysis was well-effective in answering our goal points regarding our research questions they are:

Goal 1:

Based on research question1 our goal is to find specific Time Intervals where crimes are more.

Evidence:

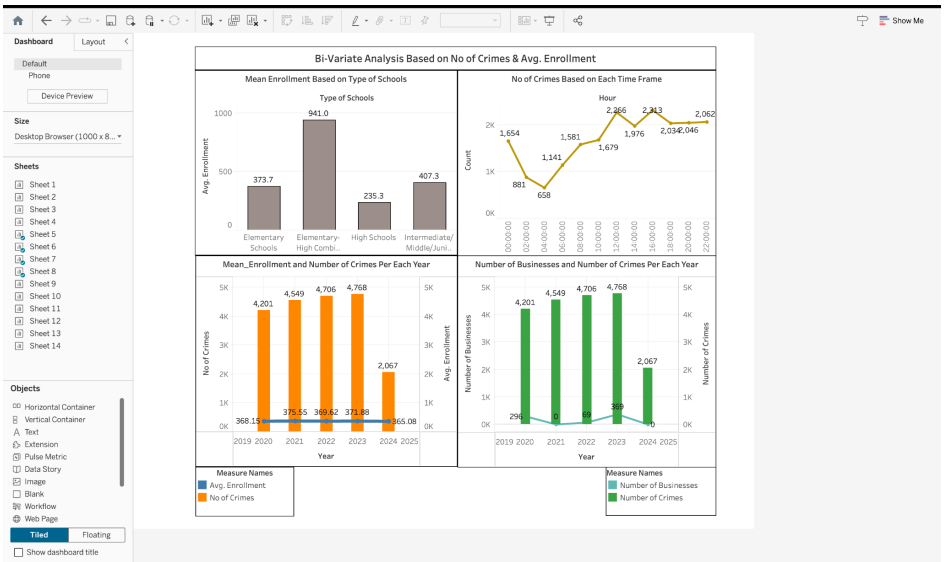

Fig. Dashboard Displaying Bivariate Analysis based crimes and Enrollment
Tool - Tableau

Answer Based on Analysis: Based on the number of crimes based on each time Frame chart in the Dashboard above, we can conclude that Time Interval '16:00-18:00 Pm' in Los Angeles is highly prone to crimes.

Goal 2:

Based on research question 2 our goal is to extract the neighbourhoods from coordinates of crime locations and categorise them to broader regions then group them based on the number of crimes.

Evidence:

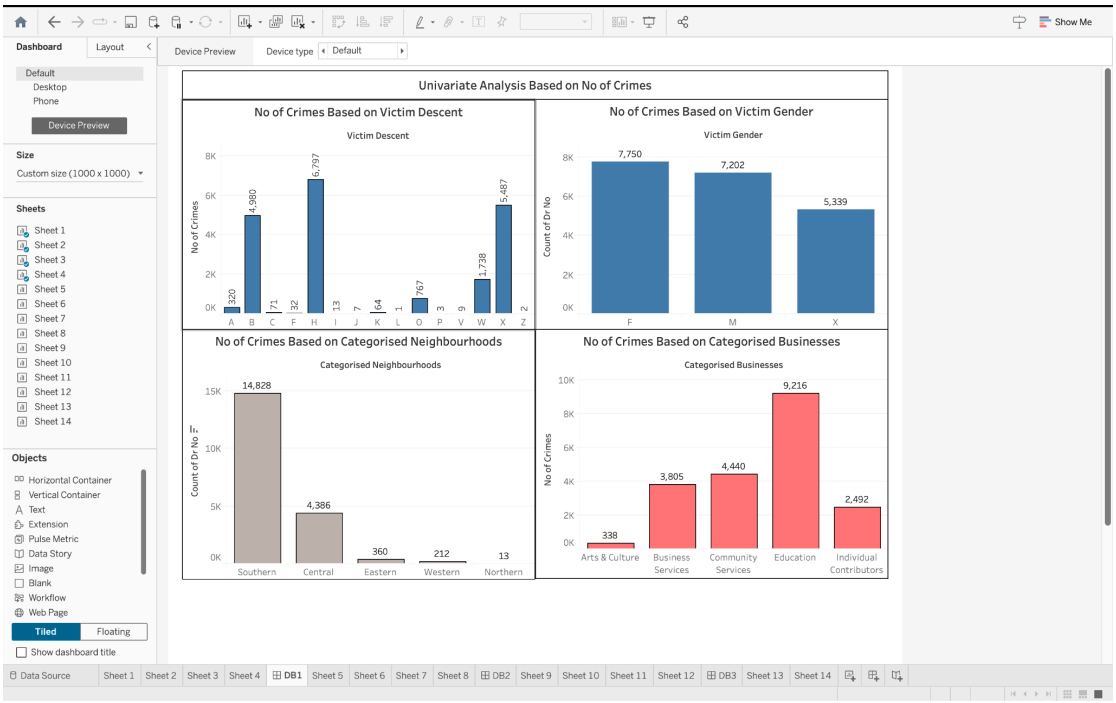

Fig. Dashboard displaying Univariate Analysis based on Number of Crimes
Tool - Tableau

Answer Based on Analysis: Considering the 'No of Crimes based on Categorised Neighbourhoods' chart from the above dashboard, we can conclude that the 'Southern' region of Los Angeles is prone to more crimes.

Goal 3:

Based on research question 3 our goal is to find the relationship between Number of Crimes, Businesses and Mean Enrollment of Schools in Each Neighbourhood and Show how crimes are effecting these facilities

Evidence:

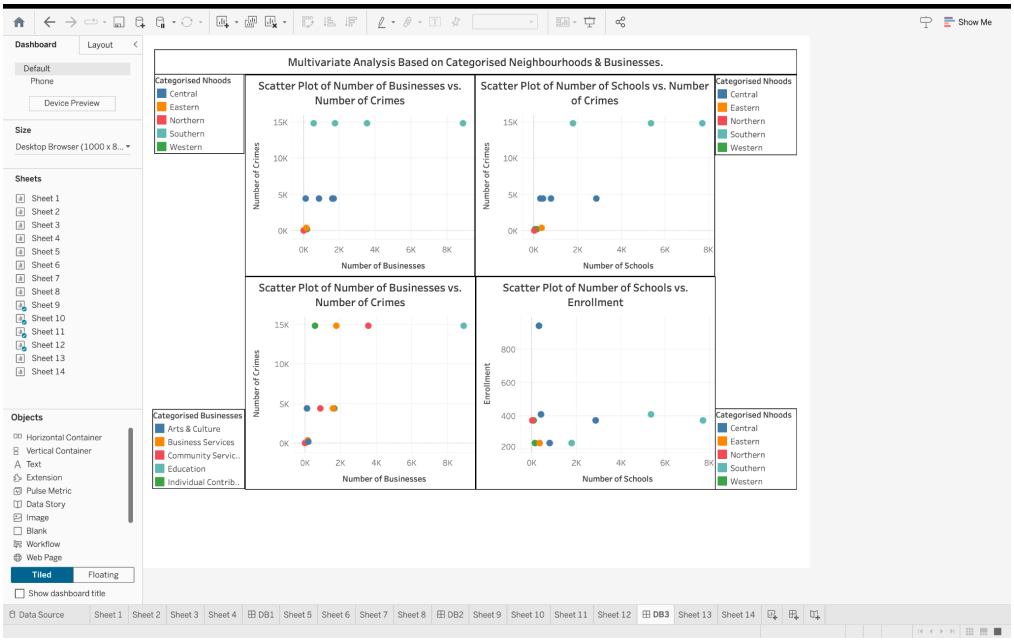

Fig. Dashboard Displaying Impact of Crimes on Facilities in each Neighborhood
Tool - Tableau

Answer Based on Analysis: Considering the 'No of schools, crimes and businesses' scatterplots from above dashboard, we can conclude that crimes and businesses and schools are directly proportional.

Goal 4:

Our goal is to predict the neighbourhood region which is categorised as high crime, low crime and medium crime

Answer Based on Analysis: Considering 'Clustering' chart in displayed in Information Modelling through k-means, we can conclude that:

- Southern Region of Los Angeles: High Crime Region
- Northern Region of Los Angeles: Low Crime Region
- Eastern Region of Los Angeles: Low Crime Region
- Western Region of Los Angeles: Low Crime Region
- Central Region of Los Angeles: Medium Crime Region

CHALLENGES

- Each dataset has area names in Los Angeles which are not properly organised, so we have to Integrate them based on extracted Neighbourhoods. Having Proper Area Names could make our analysis even more better.
- Limited Number of Data Integration methods exist till now, using joins makes the data redundant and may lead to less accuracy in the analysis.
- Robust Geocoding Techniques should be introduced to extract locations at a faster rate.
- Limited Datasets and Features are available regarding crimes, businesses and schools which made difficult to find relationships and Interpret among the three features.

LESSONS LEARNED

- The importance of having well-structured and standardized datasets became evident. Challenges in integrating area names and other inconsistencies highlighted the need for cleaner and more reliable raw data for effective analysis.
- The project emphasized the need for faster and more accurate geocoding techniques. Current methods were limited, impacting the efficiency of extracting neighborhood data based on crime coordinates.
- The restricted availability of data and features hindered deeper insights into the relationships between crimes, businesses, and schools. This highlighted the value of extensive and varied datasets for more comprehensive analysis.
- The challenges with data redundancy using joins underscored the necessity for exploring advanced integration methods that enhance accuracy without inflating redundancy.

CONCLUSION

The project successfully analyzed crime patterns in Los Angeles and identified high-crime zones, time frames, and their impact on essential services like schools and businesses. While the analysis answered the research questions, it highlighted the need for:

1. Standardized and richer datasets to enhance accuracy.
2. Advanced data integration and geocoding methods for streamlined workflows.
3. Further refinement in modeling techniques to improve predictions.

The insights from this analysis can inform public safety policies, community awareness, and business decisions, thereby contributing to sustainable urban planning.

REFERENCES

[1] Data.gov. (2024, October 4). City of Los Angeles - Crime Data from 2020 to Present. https://catalog.data.gov/dataset/crime-data-from-2020-to-present

[2] Data.gov. (2024a, September 20). City of Los Angeles - Listing of active businesses. https://catalog.data.gov/dataset/listing-of-active-businesses

[3] County of Los Angeles open data. (n.d.). https://data.lacounty.gov/datasets/32331535785b405d869ca7a7aa3abb1f/explore

[4] W3Schools.com. (n.d.). https://www.w3schools.com/python/pandas/pandas_cleaning.asp

[5] GeeksforGeeks. (2024, June 11). What is Data Visualization and Why is It Important? GeeksforGeeks. https://www.geeksforgeeks.org/data-visualization-and-its-importance/

[6] GeeksforGeeks. (2024b, September 12). Data preprocessing in data mining. GeeksforGeeks. https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/

[7] GeeksforGeeks. (2022, November 14). Google Geocoding Web Service (JSON response). GeeksforGeeks. https://www.geeksforgeeks.org/google-geo-coding-web-service-json-response/

[8] Merge, join, concatenate and compare — pandas 2.2.3 documentation. (n.d.). https://pandas.pydata.org/docs/user_guide/merging.html

[9] Ibm. (2024, October 28). Exploratory Data Analysis. *Idk*. https://www.ibm.com/topics/exploratory-data-analysis