Project Document - Group 3

Farzeem Jiwani
Agha Ahmad
Faheem Abdul Mohammed

Chicago Crime Dataset Analysis & Prediction

Index

Problem Definition

With the increase of crimes, law enforcement agencies are continuing to demand advanced geographic information systems and new data mining approaches to improve crime analytics and better protect their communities. The primary objective is to perform predictive analysis on the dataset to comprehend whether crime is a function of locality, time, climate, or any external features, along with analyzing its trends over the years. This could be helpful to law enforcement to take appropriate measures about the When-What-Where of the crime i.e. when and what type of crime could happen in what locality.

Why is this dataset more important now? Because the number of crimes in Chicago is increasing, more and more people care about their safety, and the government leader wants to build a good environment for the citizens. Therefore, it is required to predict the occurrence of a crime at a location at a specific time of a day and to anticipate if a particular neighbourhood in the city, at any given duration of the day will be a crime hotspot or not, with an acceptable rate of accuracy. Furthermore, to incorporate the impact of housing and inhabitation, literacy rate, employment, and socioeconomic status on the crime occurrence rate.

The research aims to delve deeper into this statistic and deal with missing values in our dataset. We use several graphs to analyze the dataset, and apply predictive analytics to solve some classification and prediction problems on our dataset.

Data Description

The dataset reflects reported incidents of crime (except for murders where data exists for each victim) that occurred in the City of Chicago from 2001 to the present. Data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system.
Link: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/data
Dataset Characteristics:
Size: 1.8GB
Rows: ~7.5 million
Columns: 22

Attributes Description

Attribute/ Feature	Description
Date	Best estimate Date when the incident occurred.
Block	Partially redacted address where the incident occurred.
IUCR	The Illinois Uniform Crime Reporting Code is used to classify criminal incidents when taking individual reports. Reference
Primary Type	Primary description of the IUCR code.
Description	Secondary description of the IUCR code.

Location Description	Description of the location of the incident.
Arrest	Indicates whether an arrest was made.
Domestic	Indicates whether the incident was domestic-related.
District	Indicates the police district where the incident occurred.
Community Area	Indicates the Community area where the incident occurred. Reference
Year	Year the incident occurred.
Latitude	Latitude of the location where the incident occurred.
Longitude	Longitude of the location where the incident occurred.
Location	Combination of Latitude and Longitude.

Statistics of the Data

Dataset Schema: *df.printSchema()*

```
root
 |-- ID: integer (nullable = true)
 |-- Case Number: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- Block: string (nullable = true)
 |-- IUCR: string (nullable = true)
 |-- Primary Type: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Location Description: string (nullable = true)
 |-- Arrest: boolean (nullable = true)
 |-- Domestic: boolean (nullable = true)
 |-- Beat: integer (nullable = true)
 |-- District: integer (nullable = true)
 |-- Ward: integer (nullable = true)
 |-- Community Area: integer (nullable = true)
 |-- FBI Code: string (nullable = true)
 |-- X Coordinate: integer (nullable = true)
 |-- Y Coordinate: integer (nullable = true)
 |-- Year: integer (nullable = true)
 |-- Updated On: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Location: string (nullable = true)
```

Descriptive Statistics: *df.describe().show()*

summary	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Descriptio	Beat	District	Ward	Community Are	FBI Code	X Coordinate	Y Coordinate	Year	Updated On	Latitude	Longitude
count	7404181	7404177	7404181	7404181	7404181	7404181	7404181	7395642	7404181	7404134	6789343	6790701	7404181	7330010	7330010	7404181	7404181	7330010	73300
mean	6733160.868	305161.8421	null	null	1124.836072	null	null	null	1187.538216	11.29365798	22.72210198	37.54676535	12.15570843	1164562.581	1885727.622	2009.325897	null	41.84203287	-87.671636
stddev	3333677.238	134932.5016	null	null	813.3532144	null	null	null	702.9000717	6.946559063	13.83260716	21.53874277	7.342579828	16855.58532	32282.40634	5.774777798	null	0.08881683574	0.061107009
min	634	JB299184	01/01/2001 01:00:...	0000X E 100 PL	110	ARSON	$300 AND UNDER	CTA "L" PLATFORM	111	1	1	0	01A	0	0	2001	01/01/2007 07:32:...	36.6194464	-91.686565
max	12491187	ZZZ199957	12/31/2020 12:55:...	XX UNKNOWN	9901	WEAPONS VIOLATION	WIREROOM/SPORTS	YMCA	2535	31	50	77	26	1205119	1951622	2021	12/31/2020 03:50:...	42.02291033	-87.524529

Distinct Value Count:

```
sel_columns = ['Block','IUCR','Primary Type','Location Description','District','Community Area','FBI Code','Year']

for c in sel_columns:
  print(f"Column: {c}, Unique Values Count: {df.select(c).distinct().count()}")
```

```
Column: Block, Unique Values Count: 61750
Column: IUCR, Unique Values Count: 402
Column: Primary Type, Unique Values Count: 36
Column: Location Description, Unique Values Count: 215
Column: District, Unique Values Count: 25
Column: Community Area, Unique Values Count: 79
Column: FBI Code, Unique Values Count: 26
Column: Year, Unique Values Count: 21
```

Work Distribution

Team Member	Responsibilities
Farzeem Jiwani	Pyspark, EDA, Predictive Modelling
Agha Ahmad	Pyspark SQL, Hive
Faheem Mohammed	HDFS, Hive, Project Integration

Solution Description

Abstract

We have procured criminal records from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. To accomplish our objective of recognizing crime patterns across the city based on geographical locations, we have used Hadoop and Apache Spark to achieve faster data processing and provide near real-time predictive analytics on top of that. The attributes (explained above) comprise the dataset of the system model adopted by our project and will be conducive while plotting the exact locations of the crimes.

By providing a predictive machine learning approach to determine the criminal hotspots and the location, time of committed crimes, people's awareness can be raised regarding the dangerous locations at certain times. Therefore, this project can potentially help people stay away from the locations at a certain time of the day along with saving lives.

On the other hand, police forces can use this solution to increase the level of crime prediction and prevention. Moreover, this would be useful for police resources allocation. It can help in the distribution of police at most likely crime places for any given time, to grant an efficient usage of police resources. By having all of this information available, we hope to make our community safer for the people living there and also for others who will travel there.

For a sound prediction of the occurrence of a crime at any location and any hour of a day, it is required to consider the consistent data and out of exemptions. Therefore to abstain from false conjectures and guarantee a reliable prediction model, we plan on randomizing the dataset and sampling about 6 million records to train the algorithm.

Methodology and Tools Used

Sr. No.	Task	Tools
I.	Data Storage	HDFS
II.	Exploratory Data Analysis	PySpark, PySpark SQL, Seaborn, Folium
III.	Data Querying	Hive
IV.	Data Processing	PySpark
V.	Predictive Modelling	Scikit-learn, XGBoost

1. **Data Storage:**
   We have used HDFS for data storage by uploading the Chicago Crime dataset into the File System using FileZilla and the Sandbox HDP setup.
2. **Data Exploration:**
   In this step, we have performed an extensive analysis using Pyspark and data visualization tools such as Folium and Seaborn to investigate important trends for crime detection and prevention.
3. **Data Querying**:
   In this step, we performed data analysis using Hive by creating relational tables, performing partitioning and bucketing on them to gather insights, and running performance analysis on the dataset.
4. **Data Processing:**
   a. Dropped missing/null values that accounted for <1% of data
   b. Excluded records with vague information
   c. Removed extremely rare crime types and locations
   d. Parsed the date column and broke it into components (year, month, day, hour, minute)
   e. Factorized the dataset
   f. Filter out irrelevant features from the dataset using correlation
   g. Used Random sampling techniques to balance the data
5. **Predictive Modelling:**
   Implemented Random Forest and XGBoost algorithm as proof-of-concept models to run predictions by dividing the dataset into training and testing.

How the Tools were used

- **HDFS**
  It was used as a storage tool to persist the input Crime dataset.
- **Pyspark & Pyspark SQL**
  - Pyspark data frames were used to read the CSV input into a well-defined relational schema using SQL types.
  - Pyspark functions such as *groupBy* were used to group on fundamental attributes such as *Primary Type* and *Location* of the crime and perform aggregations on top of them.
- **Hive**
  - Hive was used to create a well-defined relational database.
  - Hive partitioning and bucketing was used to create 3 data tables for the crime dataset
  - Perform execution time comparison on partitioning, bucketing, and normal data tables
- **Scikit-learn, Folium, Seaborn**
  - Scikit-learn was used to import the required ML libraries to implement Random Forest and XGBoost models.
  - Folium was used to draw interactive maps using *Latitude* and *Longitude* while Seaborn was used for insightful data visualizations.

Why they were chosen

- **HDFS**
  HDFS was chosen due to its reliable, fault-tolerant, and highly available architecture. Since we are dealing with ~7 million records, this design suits the best for storing such an amount of data.

- **Pyspark & Pyspark SQL**
  - Due to its speed for large data processing.
  - Easy built-in APIs and Functions used for reading and operating on the Crime dataset.
  - Built-in support for running SQL queries on the columnar data and providing insights used for visualizations.

- **Hive**
  - Ease to load and process structured data.
  - Well equipped to provide data summarization and analysis in a few queries.
  - Scalable for large datasets such as ours without any performance issues.
  - Partitioning and bucketing of the data improve overall query performance.
  - Provides an efficient environment for ETL.

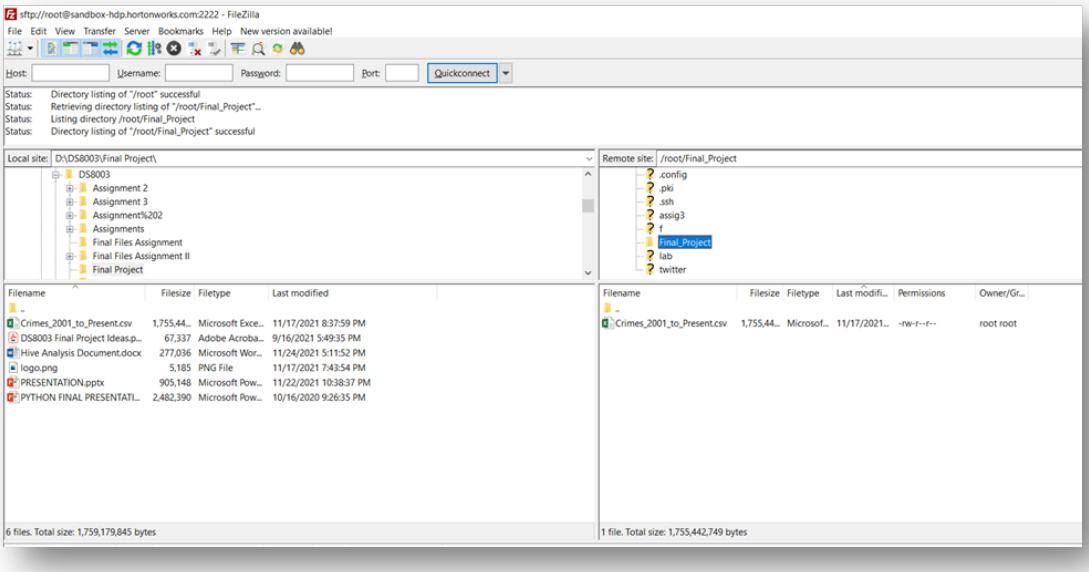
- **Scikit-learn**
  - Extensive support for supervised learning algorithms.

Data Preparation

1. Downloading the Crimes_2001_to_Present.csv
2. Creating a Project Folder to load the dataset into it.

```
[root@sandbox-hdp ~]# mkdir Final_Project
```

3. Load datasets from node to Unix by using Filezilla



4. Creating the Project Folder on HDFS:

```
[root@sandbox-hdp Final_Project]# hadoop fs –mkdir /user/root/Project
```

5. Copying the Crimes_2001_to_Present.csv to hdfs:

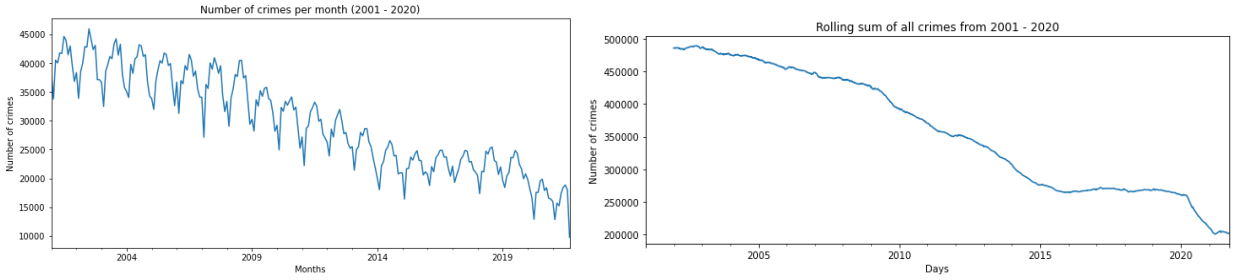
```
[root@sandbox-hdp Final_Project]# hadoop fs -put
Crimes_2001_to_Present.csv /user/root/Project/

[root@sandbox-hdp Final_Project]# hadoop fs -ls /user/root/Project
```

```
[root@sandbox-hdp Final_Project]# hadoop fs -mkdir /user/root/Project
[root@sandbox-hdp Final_Project]# hadoop fs -put Crimes_2001_to_Present.csv /user/root/Project/
[root@sandbox-hdp Final_Project]# hadoop fs -ls /user/root/Project
Found 1 items
-rw-r--r--   1 root root 1755442749 2021-11-29 20:22 /user/root/Project/Crimes_2001_to_Present.csv
[root@sandbox-hdp Final_Project]#
```

Describing Insights

Crime Trends over the years

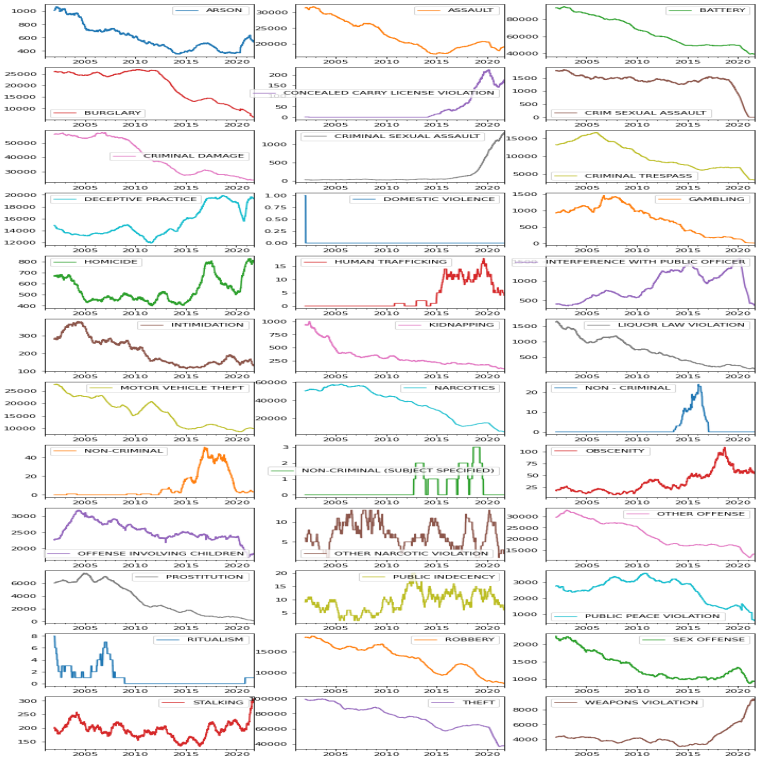
```
crimes.resample('M').size().plot(legend=False)
crimes.resample('D').size().rolling(365).sum().plot()
```



The above two graphs show a decreasing pattern of crime over the years.

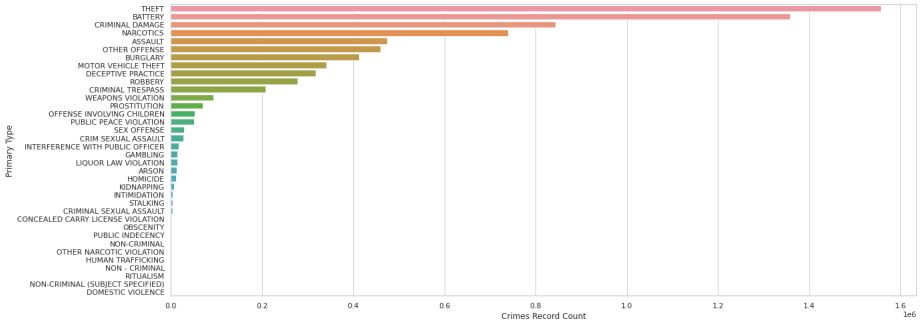
```
crimes_count_date = crimes.pivot_table('ID', aggfunc=np.size, columns='Primary Type', index=crimes.index.date, fill_value=0)
crimes_count_date.index = pd.DatetimeIndex(crimes_count_date.index)
plo = crimes_count_date.rolling(365).sum().plot(figsize=(12, 30), subplots=True, layout=(-1, 3), sharex=False, sharey=False)
```

However, on further analysis for each crime type, it was found not to be true as shown below.



Crimes based on Primary Type

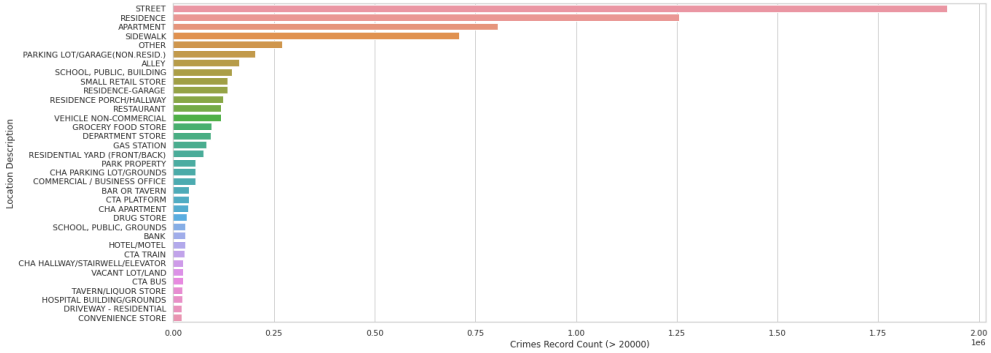
```
crime_type_groups = df.groupBy('Primary Type').count()
crime_type_counts = crime_type_groups.orderBy('count',ascending=False)
```



The above graph depicts the distribution of crimes based on their *Primary Type*, among which, most crimes are *Theft* and *Battery*.

Crimes based on Location

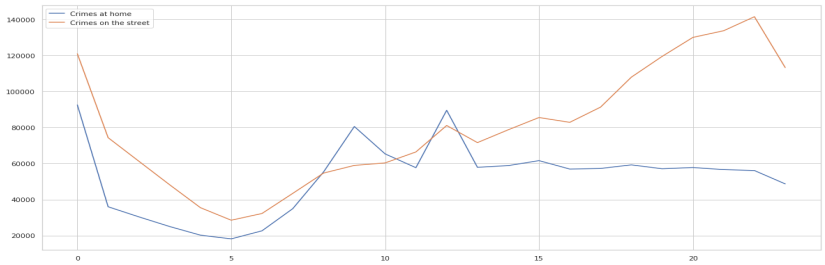
```
crime_loc_groups = df.groupBy('Location Description').count()
crime_loc_counts = crime_loc_groups.orderBy('count',ascending=False)
```



The above graph depicts the distribution of crimes based on their *Location*, among which, most crimes occur on the *Street* or *Residence*.

The below graph represents Crimes at Residence and on the Street affected over an hour of the day.

```
street_home_hour = location_hour.where((location_hour['Location Description']
== 'STREET') | (location_hour['Location Description'] == 'RESIDENCE'))
```



It seems that crimes at home increase when crimes on the street decrease. Could it be that it's the same kind of people that are in charge of both home and streets?

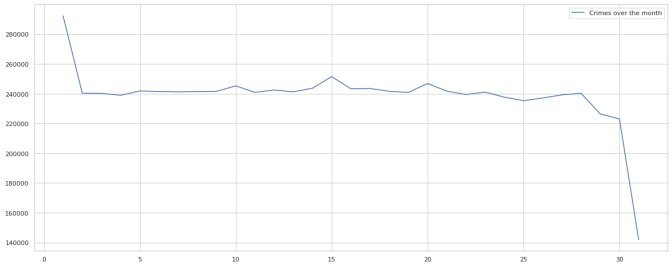
Monthly and Weekly Crime Analysis

```
df_dates = df_hour.withColumn('week_day', dayofweek(df_hour['date_time']))\
                  .withColumn('year_month', month(df_hour['date_time']))\
                  .withColumn('month_day', dayofmonth(df_hour['date_time']))\
                  .withColumn('date_number', datediff(df['date_time'],
to_date(lit('2001-01-01'), format='yyyy-MM-dd')))\
                  .cache()
```

Crimes over a day of the month

```
month_day_crime_counts = df_dates.groupBy('month_day').count()
```

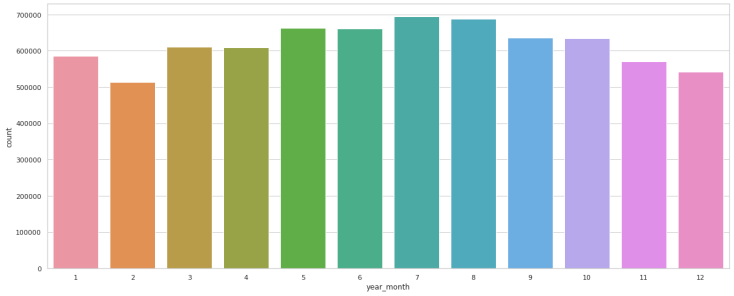
It can be seen that crimes are constant over the month days with a slight decrease towards the start and end of the month days.



Crimes over a month of a year

```
year_month_crime_counts = df_dates.groupBy('year_month').count()
```

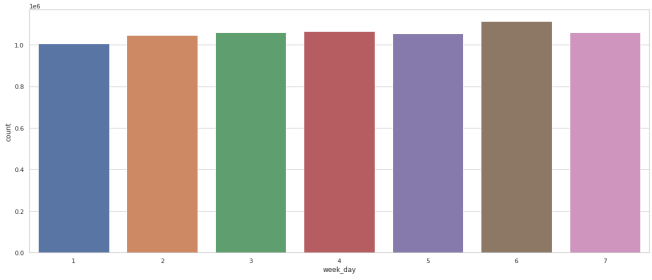
The figure illustrates that months July and August have the highest crime rate.



Crimes over a day of a week

```
week_day_crime_counts = df_dates.groupBy('week_day').count()
```

The below graph depicts that crime rates are steady over the week with a gradual increase over the weekend.

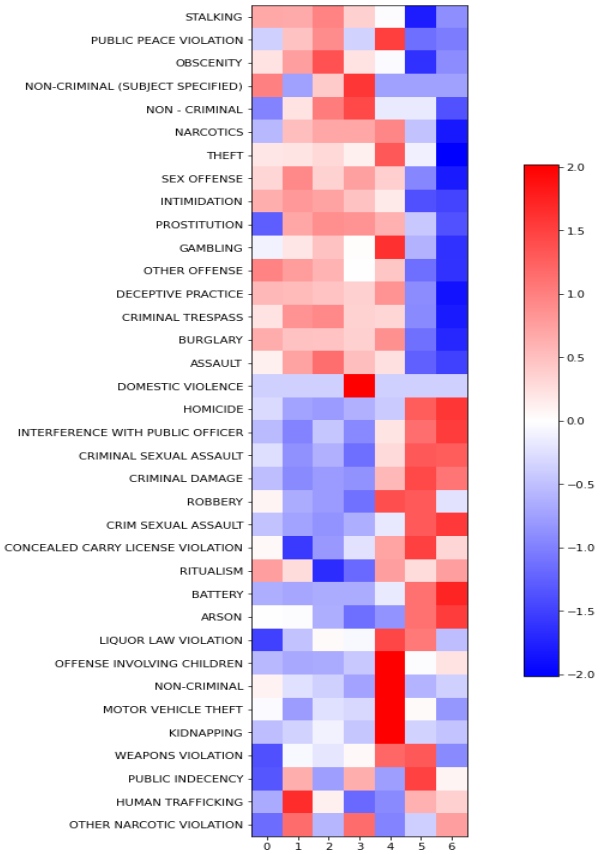


Code Snippet for heatmaps using pivot tables:

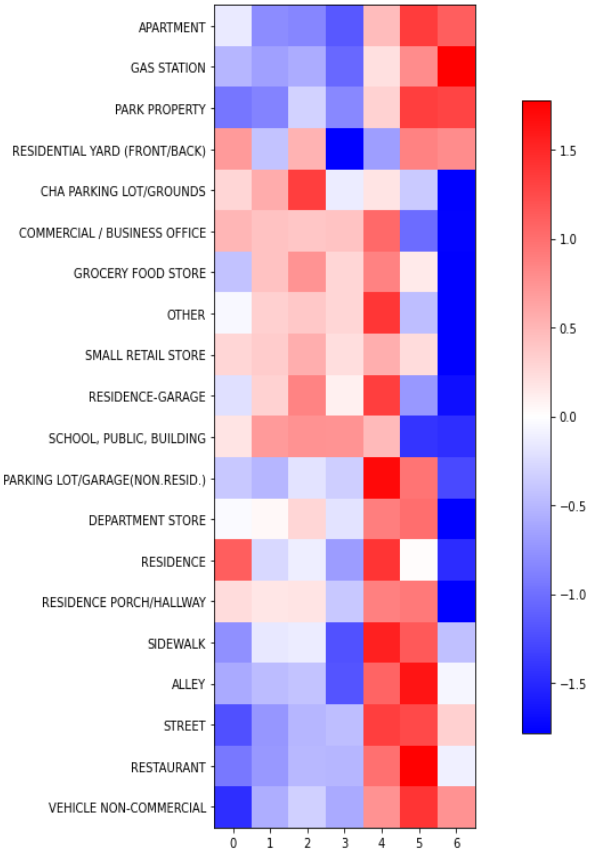
```
hour_by_location = crimes.pivot_table(values='ID', index='Location Description', columns=crimes.index.hour, aggfunc=np.size).fillna(0)
hour_by_type     = crimes.pivot_table(values='ID', index='Primary Type', columns=crimes.index.hour, aggfunc=np.size).fillna(0)
hour_by_week     = crimes.pivot_table(values='ID', index=crimes.index.hour, columns=crimes.index.day_name(), aggfunc=np.size).fillna(0)
hour_by_week     = hour_by_week[days].T # just reorder columns according to the the order of days
dayofweek_by_location = crimes.pivot_table(values='ID', index='Location Description', columns=crimes.index.dayofweek, aggfunc=np.size).fillna(0)
dayofweek_by_type = crimes.pivot_table(values='ID', index='Primary Type', columns=crimes.index.dayofweek, aggfunc=np.size).fillna(0)
location_by_type  = crimes.pivot_table(values='ID', index='Location Description', columns='Primary Type', aggfunc=np.size).fillna(0)
```

```
from sklearn.cluster import AgglomerativeClustering as AC

def scale_df(df,axis=0):
    '''
    A utility function to scale numerical values (z-scale) to have a mean of zero
    and a unit variance.
    '''
    return (df - df.mean(axis=axis)) / df.std(axis=axis)

def plot_hmap(df, ix=None, cmap='bwr'):
    '''
    A function to plot heatmaps that show temporal patterns
    '''
    if ix is None:
        ix = np.arange(df.shape[0])
    plt.imshow(df.iloc[ix,:], cmap=cmap)
    plt.colorbar(fraction=0.03)
    plt.yticks(np.arange(df.shape[0]), df.index[ix])
    plt.xticks(np.arange(df.shape[1]))
    plt.grid(False)
    plt.show()

def scale_and_plot(df, ix = None):
    '''
    A wrapper function to calculate the scaled values within each row of df and plot_hmap
    '''
    df_marginal_scaled = scale_df(df.T).T
    if ix is None:
        ix = AC(4).fit(df_marginal_scaled).labels_.argsort() # a trick to make better heatmaps
    cap = np.min([np.max(df_marginal_scaled.values), np.abs(np.min(df_marginal_scaled.values))])
    df_marginal_scaled = np.clip(df_marginal_scaled, -1*cap, cap)
    plot_hmap(df_marginal_scaled, ix=ix)
```

Day of the week by Primary type of the crime



Day of the week vs Crime Location



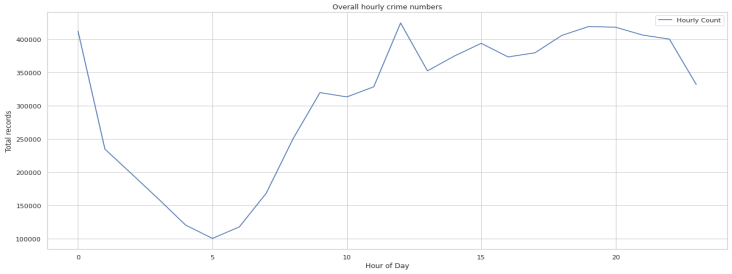
Some crime types peak at specific days of the week (like Friday or Saturday). Other types are more like weekday crimes (showing in the mid-part of the left heatmap) or weekend crimes (showing in the lower part of the left heatmap).

Similarly, some crimes occur at specific locations during particular days of the week (right heatmap).

Hourly Crime Analysis

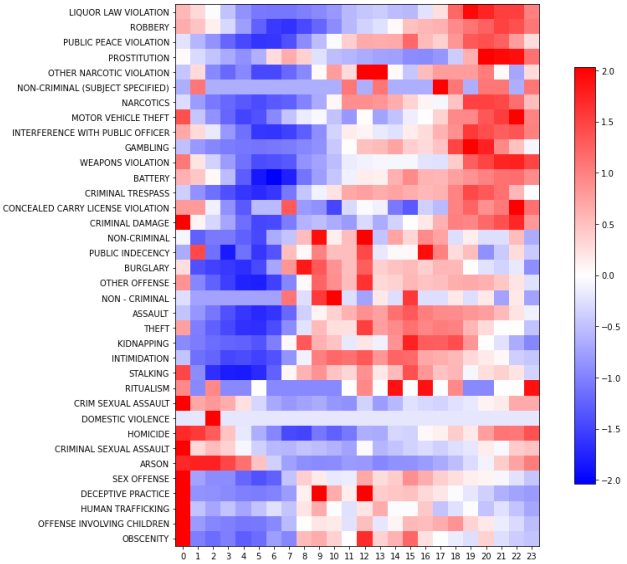
Total Crimes over an hour of the day

It seems that hours 17-22 have the most number of crimes.



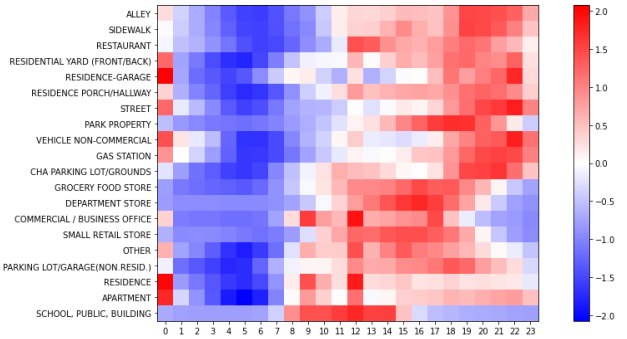
Heatmap of Hour of the day vs Primary Type of the Crime

As you can see, some crimes have their peaks during the early morning (upper part of the heatmap), other crimes peak during the day (mid-part of the heatmap), and the final group of crimes peak at night (lower part of the heatmap).



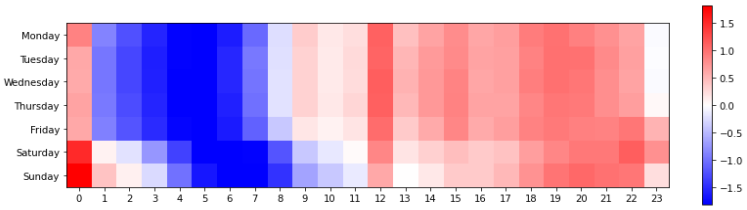
Heatmap of Hour of the day vs Location of the Crime

The upper part of the heatmap shows locations that have crime peaks during the day (with a couple of them peaking between 9 AM - 12 PM). The mid-part shows locations that have crime peaks during the night and finally the lower part shows that bars have crime peaks during the early morning.



Heatmap of Hour of the day vs Day of the week

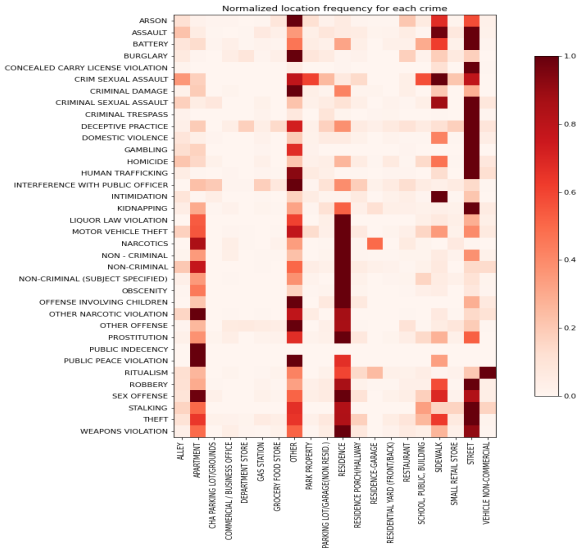
As you can see, weekends show a different pattern concerning residences, concerning the above crimes.



Location vs Primary Type Crime Frequency Analysis

```
df = normalize(location_by_type)
ix = AC(3).fit(df.T).labels_.argsort() # a trick to make better heatmaps
plt.figure(figsize=(17,13))
plt.imshow(df.T.iloc[ix,:], cmap='Reds')
plt.colorbar(fraction=0.03)
plt.xticks(np.arange(df.shape[0]), df.index, rotation='vertical')
plt.yticks(np.arange(df.shape[1]), df.columns)
plt.title('Normalized location frequency for each crime')
plt.grid(False)
plt.show()
```

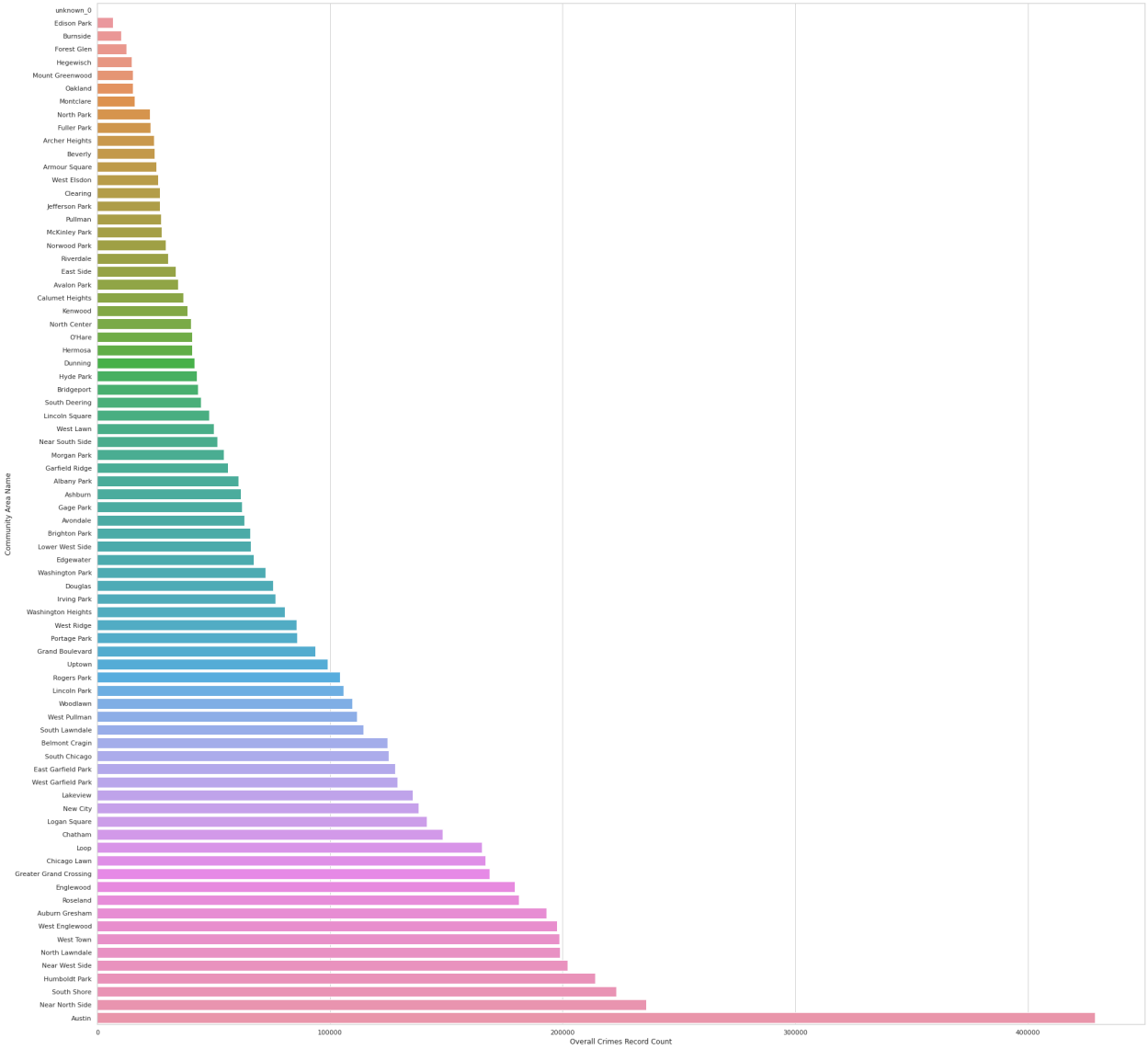
In this analysis, as with the previous ones, we take each crime type and re-normalize its location frequency to be between zero and one. Using Agglomerative Clustering, we can look at the top frequent locations of each crime type (darker red reflects a more frequent location).



Well, it appears that most crimes occur at either apartments, residences, sidewalks, streets, or 'other'. There are few notable exceptions here and there but those places are very frequent.

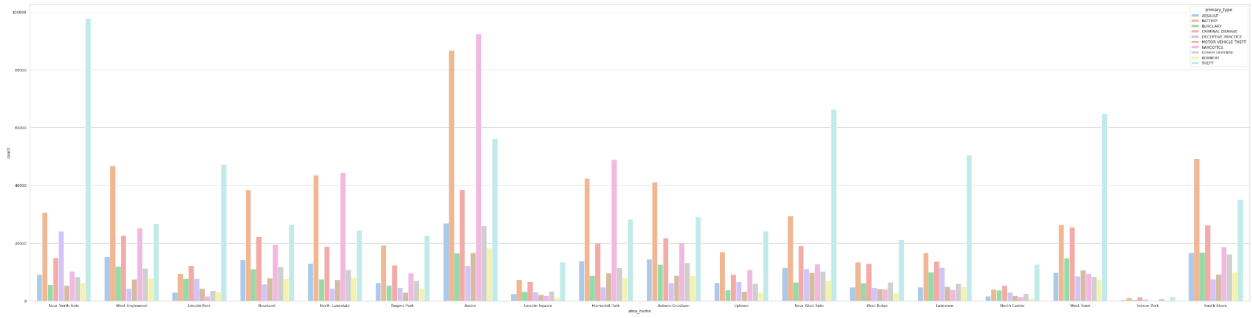
Crimes - Community Area Analysis

```
community_area_counts['community_area_name'] = community_area_counts['Community Area'].apply(lambda area: area_name_dic.get(float(area),  'unknown_%s'%area))
community_area_counts = community_area_counts.sort_values(by='count')
```



It seems that Austin has a drastically higher crime rate than the rest.

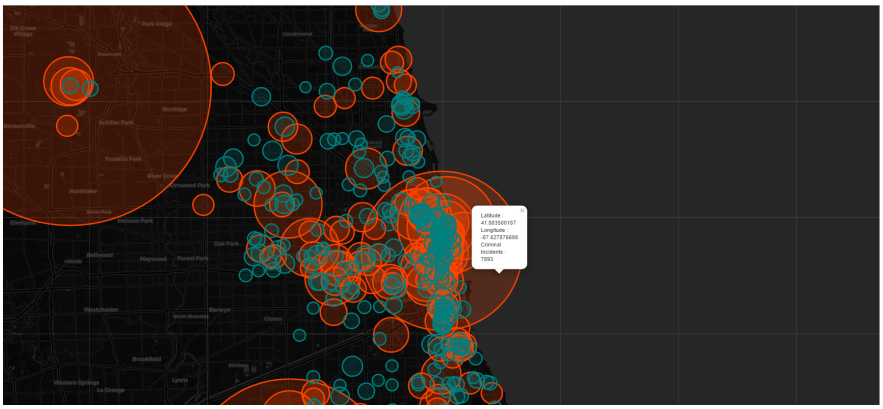
```
code_pairs_df = spark.createDataFrame(code_pairs, ['community_area', 'area_name'])
named_tops_of_tops = code_pairs_df.join(tops_of_tops, on='community_area', how='right')
tops_of_tops_dff = pd.DataFrame(named_tops_of_tops.rdd.map(lambda l: l.asDict()).collect() )
```



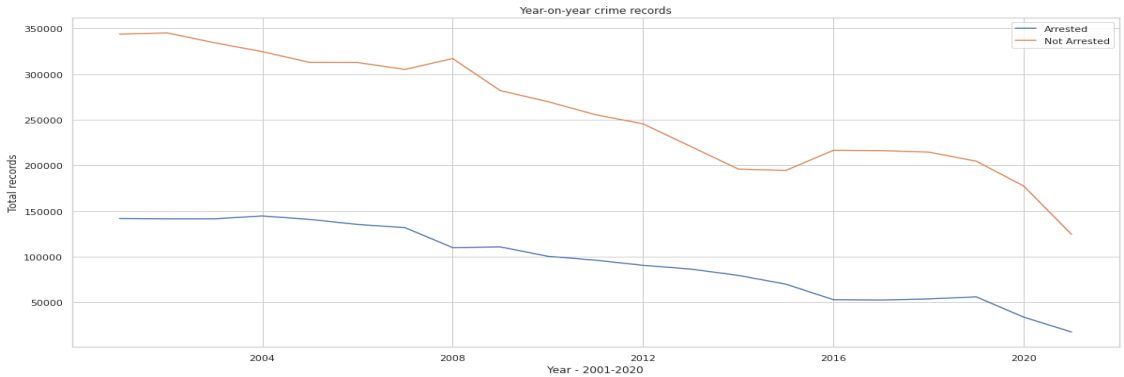
The above graph outlines the top 10 crime types in various community areas with peak crimes occurring in Austin.

Crime Hotspot Map

The below is an interactive map created using Folium that illustrates the number of criminal incidents that occurred at a given latitude and longitude.



Arrests over the years



The above graph shows how the Arrests have evolved over the 20 years. It looks like the relative distance between arrests and non-arrests has remained constant.

Hive Query Analysis

Table Structure - Partitioning on attribute Primary Type

```
hive> dfs -ls /apps/hive/warehouse/chicago_crimes.db/crime_part;
Found 38 items
drwxr-xr-x   - root hadoop          0 2021-11-24 02:50 /apps/hive/warehouse/chicago_crimes.db/crime_part/.hive-staging_hive_2021-11-24_02-26-52_091_4443651590611713320
1
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=ARSON
drwxrwxrwx   - root hadoop          0 2021-11-24 04:00 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=ASSAULT
drwxrwxrwx   - root hadoop          0 2021-11-24 04:00 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=BATTERY
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=BURGLARY
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=CONCEALED CARRY LICENSE VIOLATION
drwxrwxrwx   - root hadoop          0 2021-11-24 04:00 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=CRIM SEXUAL ASSAULT
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=CRIMINAL DAMAGE
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=CRIMINAL SEXUAL ASSAULT
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=CRIMINAL TRESPASS
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=DECEPTIVE PRACTICE
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=DOMESTIC VIOLENCE
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=GAMBLING
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=HOMICIDE
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=HUMAN TRAFFICKING
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=INTERFERENCE WITH PUBLIC OFFICER
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=INTIMIDATION
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=KIDNAPPING
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=LIQUOR LAW VIOLATION
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=MOTOR VEHICLE THEFT
drwxrwxrwx   - root hadoop          0 2021-11-24 04:00 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=NARCOTICS
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=NON - CRIMINAL
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=NON-CRIMINAL
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=NON-CRIMINAL (SUBJECT SPECIFIED)
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=OBSCENITY
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=OFFENSE INVOLVING CHILDREN
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=OTHER NARCOTIC VIOLATION
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=OTHER OFFENSE
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=PROSTITUTION
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=PUBLIC INDECENCY
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=PUBLIC PEACE VIOLATION
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=Primary Type
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=RITUALISM
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=ROBBERY
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=SEX OFFENSE
drwxrwxrwx   - root hadoop          0 2021-11-24 03:59 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=STALKING
drwxrwxrwx   - root hadoop          0 2021-11-24 04:00 /apps/hive/warehouse/chicago_crimes.db/crime_part/primarytype=THEFT
```

Table Structure - Bucketing on attribute Year

```
hive> dfs -ls /apps/hive/warehouse/chicago_crimes.db/crime_bucket;
Found 20 items
-rwxrwxrwx   1 root hadoop   82083314 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000000_0
-rwxrwxrwx   1 root hadoop  132610656 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000001_0
-rwxrwxrwx   1 root hadoop   96411543 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000002_0
-rwxrwxrwx   1 root hadoop   94913752 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000003_0
-rwxrwxrwx   1 root hadoop   93677747 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000004_0
-rwxrwxrwx   1 root hadoop   90524228 2021-11-24 03:31 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000005_0
-rwxrwxrwx   1 root hadoop   89690211 2021-11-24 03:31 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000006_0
-rwxrwxrwx   1 root hadoop   87675998 2021-11-24 03:31 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000007_0
-rwxrwxrwx   1 root hadoop   85381550 2021-11-24 03:31 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000008_0
-rwxrwxrwx   1 root hadoop   78881454 2021-11-24 03:31 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000009_0
-rwxrwxrwx   1 root hadoop   74418116 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000010_0
-rwxrwxrwx   1 root hadoop   70846272 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000011_0
-rwxrwxrwx   1 root hadoop   68065151 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000012_0
-rwxrwxrwx   1 root hadoop   62126177 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000013_0
-rwxrwxrwx   1 root hadoop   55924358 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000014_0
-rwxrwxrwx   1 root hadoop   53941382 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000015_0
-rwxrwxrwx   1 root hadoop   55186921 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000016_0
-rwxrwxrwx   1 root hadoop   55096472 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000017_0
-rwxrwxrwx   1 root hadoop   54923107 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000018_0
-rwxrwxrwx   1 root hadoop   53617219 2021-11-24 03:32 /apps/hive/warehouse/chicago_crimes.db/crime_bucket/000019_0
hive>
```

Districts with the most reported incidents

Tables	Normal	Partition	Bucketing
Query	SELECT district, COUNT(*) AS cntdistrict FROM chicago_crimes.crime GROUP BY district ORDER BY cntdistrict DESC	SELECT district, COUNT(*) AS cntdistrict FROM chicago_crimes.crime_part GROUP BY district ORDER BY cntdistrict DESC	SELECT district, COUNT(*) AS cntdistrict FROM chicago_crimes.crime_bucket GROUP BY district ORDER BY cntdistrict DESC
Execution Time	43.478	39.729	29.22

Rows returned: 33

8	483013
11	465965
7	422495
6	420187
25	412906
4	401801
3	366203
9	356144
12	348159
2	338125
18	325709
19	325337
5	316648
15	313377
10	309642
1	289103
14	283447
NULL	240421
16	238526
22	229699
24	218987
17	206496
20	125176
31	216
122	131
113	106
111	33
121	19
124	12
123	8
112	5
21	4
114	4

Arrests by primary type

Tables	Normal	Partition	Bucketing
Query	SELECT primarytype, COUNT(*) AS cnt FROM chicago_crimes.crime WHERE arrest = True GROUP BY primarytype ORDER BY cnt DESC	SELECT primarytype, COUNT(*) AS cnt FROM chicago_crimes.crime_part WHERE arrest = True GROUP BY primarytype ORDER BY cnt DESC	SELECT primarytype, COUNT(*) AS cnt FROM chicago_crimes.crime_bucket WHERE arrest = True GROUP BY primarytype ORDER BY cnt DESC
Execution Time	76.652	72.245	65.151

Rows returned: 36

```
NARCOTICS       723119
BATTERY 283951
THEFT   178618
CRIMINAL TRESPASS       144350
ASSAULT 96873
OTHER OFFENSE   81290
PROSTITUTION    69161
WEAPONS VIOLATION       68585
CRIMINAL DAMAGE 57013
DECEPTIVE PRACTICE      44545
PUBLIC PEACE VIOLATION  29890
MOTOR VEHICLE THEFT     27193
ROBBERY 25967
BURGLARY        23092
INTERFERENCE WITH PUBLIC OFFICER        16271
GAMBLING        14273
LIQUOR LAW VIOLATION    14066
OFFENSE INVOLVING CHILDREN      10500
SEX OFFENSE     7745
HOMICIDE        5407
CRIM SEXUAL ASSAULT     4334
ARSON   1468
CONCEALED CARRY LICENSE VIOLATION       801
KIDNAPPING      744
INTIMIDATION    643
STALKING        604
OBSCENITY       536
CRIMINAL SEXUAL ASSAULT 317
PUBLIC INDECENCY        178
OTHER NARCOTIC VIOLATION        95
NON-CRIMINAL    11
HUMAN TRAFFICKING       6
NON - CRIMINAL  6
NON-CRIMINAL (SUBJECT SPECIFIED)        3
RITUALISM       3
DOMESTIC VIOLENCE       1
```

Predictive Modelling

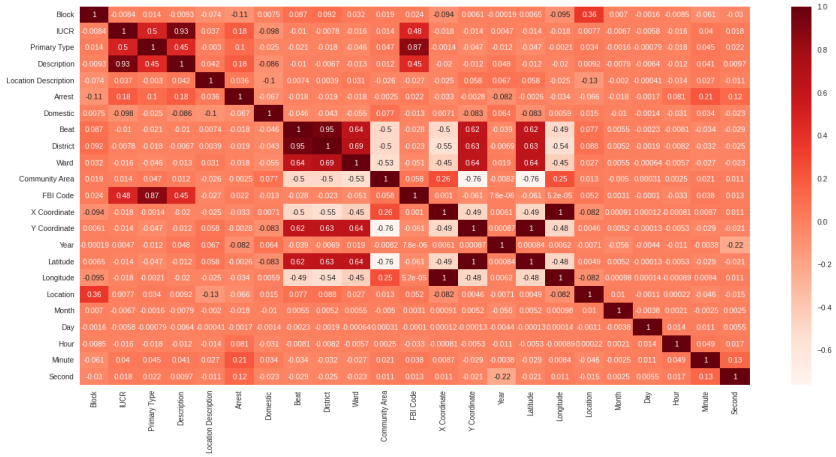
Feature Selection

Based on the below correlation matrix, highly relevant features to *Primary Type* were selected such as *['Block', 'IUCR', 'Description', 'Location Description',*

'Arrest', 'Domestic', 'Beat', 'District', 'Ward', 'Community Area',

'X Coordinate', 'Y Coordinate', 'Year', 'Latitude',

'Longitude', 'Location', 'Month', 'Day', 'Hour', 'Minute', 'Second']



Model Building & Evaluation

Below are the two models built with their respective hyperparameters, classification report and evaluation metrics:

Future Work

For future work, to boost the classification accuracy, it will be necessary to incorporate other datasets such as the police departmental and demographic information. For example, the police department may focus on solving a specific type of crime during a specific period, which may reduce the occurrence of that type of crime. Additionally, some events and the outcomes of the events may be associated with some crime types, for example, basketball games, baseball games, and elections. Weather information and classification of buildings can also be incorporated.

The end goal would be to create a web application portal that receives an address or location and time arguments in Chicago from the user and predicts the probability of crime (hence safety) at a specific location and a particular time.

In terms of lessons learnt, we have found that Hive is more semantically friendly and gives your dataset a well-defined relational schema while Spark is much more powerful for data cleaning and processing. In conclusion, we have learnt and implemented the tools taught in the course to the best of our knowledge.

References

1. CLEAR dataset by Chicago police department https://data.cityofchicago.org/public-safety/crimes-2001-to-present/ijzp-q8t2.

2. S. Yadav, M. Timbadia, A. Yadav, R. Vishwakarma, and N. Yadav, "Crime pattern detection, analysis & prediction," 2017 International conference of Electronics, Communication, and Aerospace Technology (ICECA), Coimbatore, 2017, pp. 225-230.

3. A. Bogomolov, B. Lepri, J. Staiano, N. Oliver, F. Pianesi and A. Pentland, 'Once Upon a Crime: Towards Crime Prediction from Demographics and Mobile Data', CoRR, vol. 14092983, 2014.

4. R. Arulanandam, B. Savarimuthu, and M. Purvis, 'Extracting Crime Information from Online Newspaper Articles', in Proceedings of the Second Australasian Web Conference — Volume 155, Auckland, New Zealand, 2014, pp. 31–38.

5. Hsinchun Chen, Wingyan Chung, Jennifer Jie Xu, Gang Wang, Yi Qin, and Michael Chau. Crime data mining: a general framework and some examples. computer, 37(4):50–56, 2004.

6. Matthew S Gerber. Predicting crime using Twitter and kernel density estimation. Decision Support Systems, 61:115–125, 2014.

7. T. Almanie, R. Mirza, and E. Lor, "Crime prediction based on crime types and using spatial and temporal criminal hotspots," arXiv preprint arXiv:1508.02050, 2015.