– Crime Analysis –

Mohamed Sameh Ahmed : 17p3033

Nour Eldin Talaat : 18p3826

Reda Mohsen Reda : 18p5141

Nour Eldin Khaled : 18p5722

Youssef Massoud : 18p8814

(For contact us: 18p5141@eng.asu.edu.eg / 01117532355)

Link GitHub:

reda-mohsen/Crime-Analysis (github.com)

Introduction

The high prevalence of crime in modern times presents a significant problem for a city's police force. The various law enforcement groups have amassed a tremendous amount of data over the past few decades. Data offers information about several forms of criminal episodes that have occurred in various states and cities across a particular country, rather than just one specific type of crime. The proper method of bringing the perpetrator to justice has been to solve the offences. Additionally, data presents both more opportunities and challenges for researchers and analysts.

Data analytics can discover connections between the data's features so they can help the police track down the culprit. Finding trends and distinctive patterns in crime report analysis requires a different methodology. Finding out what kind of crime will occur where and when is the primary goal of crime prediction, which is a key task in criminal analysis. The process of predicting crime becomes challenging and complex, especially when information about the criminal incident is included, such as the criminal's profile, social network of operations, and any geographic information, such as the time of the occurrence and the location of the illegal action, also gets included in the analysis.

Problem Definition

Finding out what kind of crime will occur where and when is the primary goal of crime prediction. Find the criminals involved in the incident. Identifying can be accomplished by employing big data analytics to analyze the data that has been collected from the crime scene by the researchers and analyzers. To make the best use of technical and human resources, it is crucial to identify the suspects in situations.

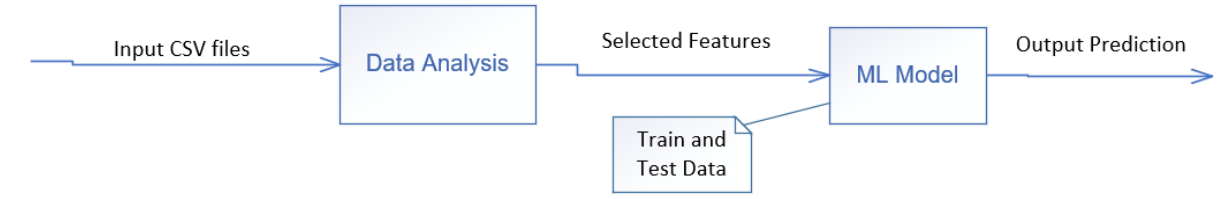
The security forces are employing techniques for crime-analysis in order to resolve such instances.

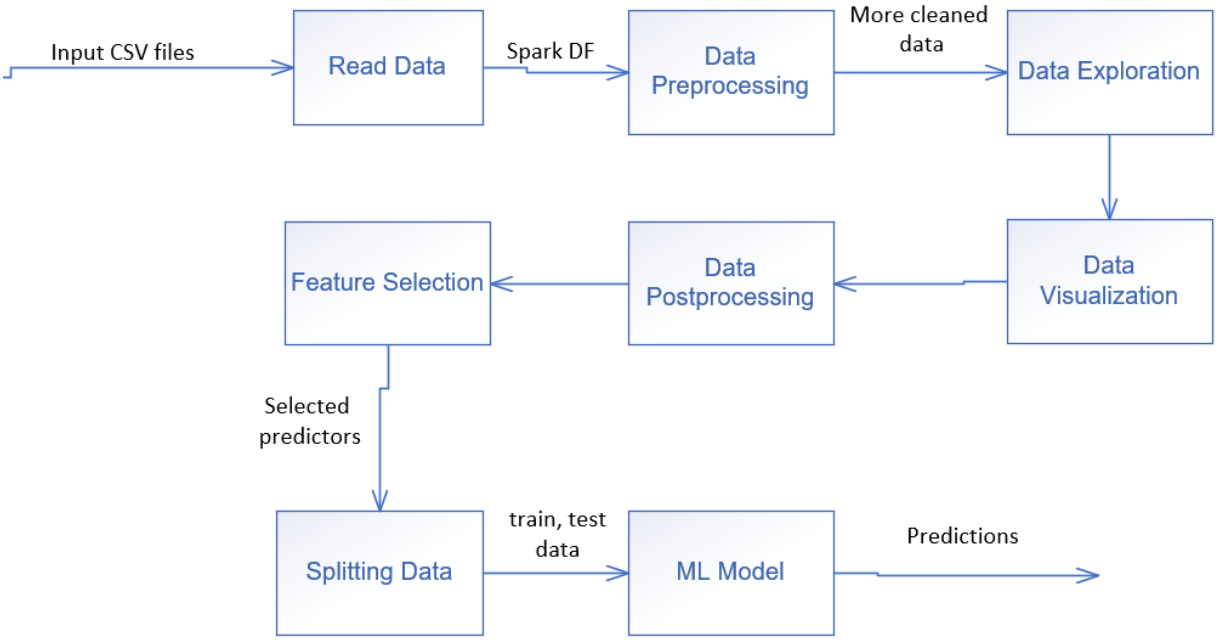
Problem Solution

Spark framework and python programming are used to forecast the sort of crime that will occur in a specific location as well as to determine crime trends. The categorical nature of crime data makes it simple to apply analysis techniques to the dataset. Using spark machine learning, the suggested framework is evaluated for crime prediction in an effort to prevent future incidents of the same crime from occurring in the same location and to identify the offender.

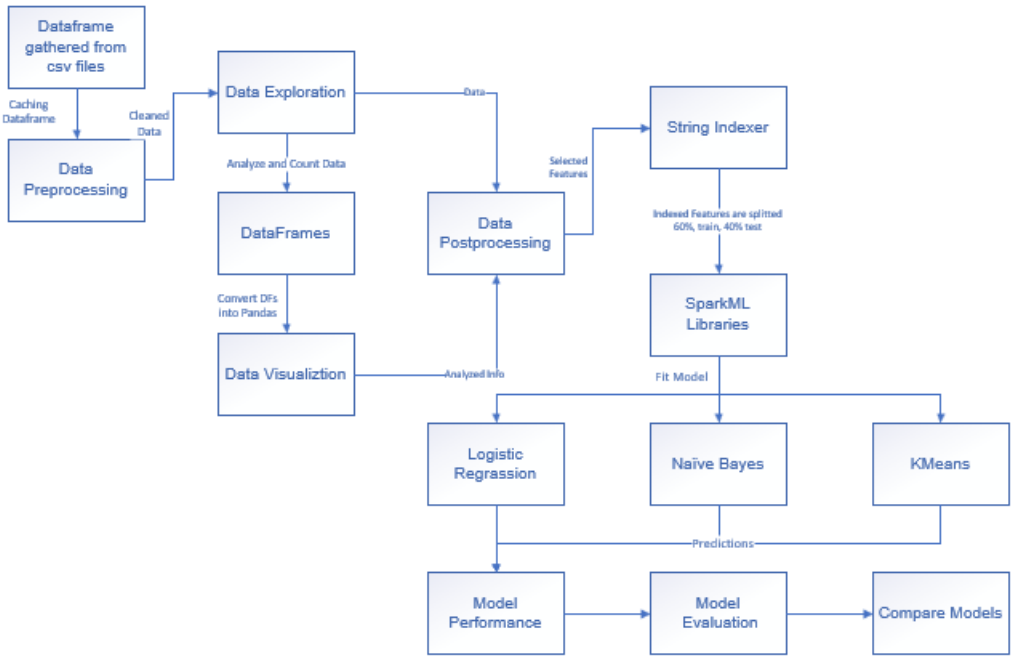
Diagrams

Block Diagrams

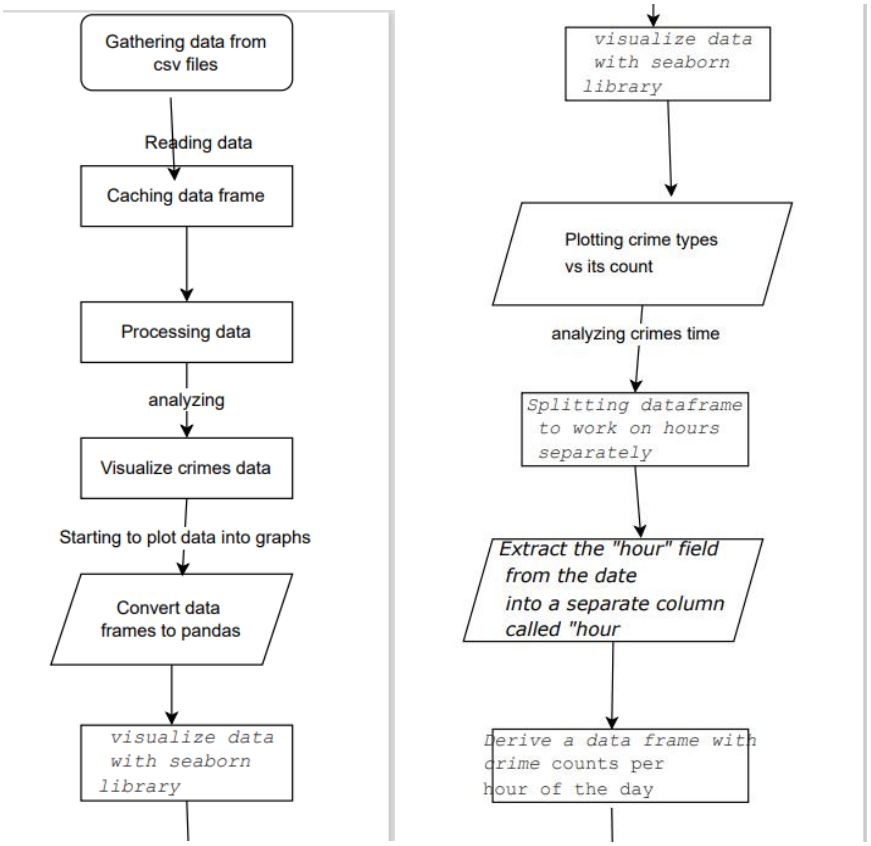


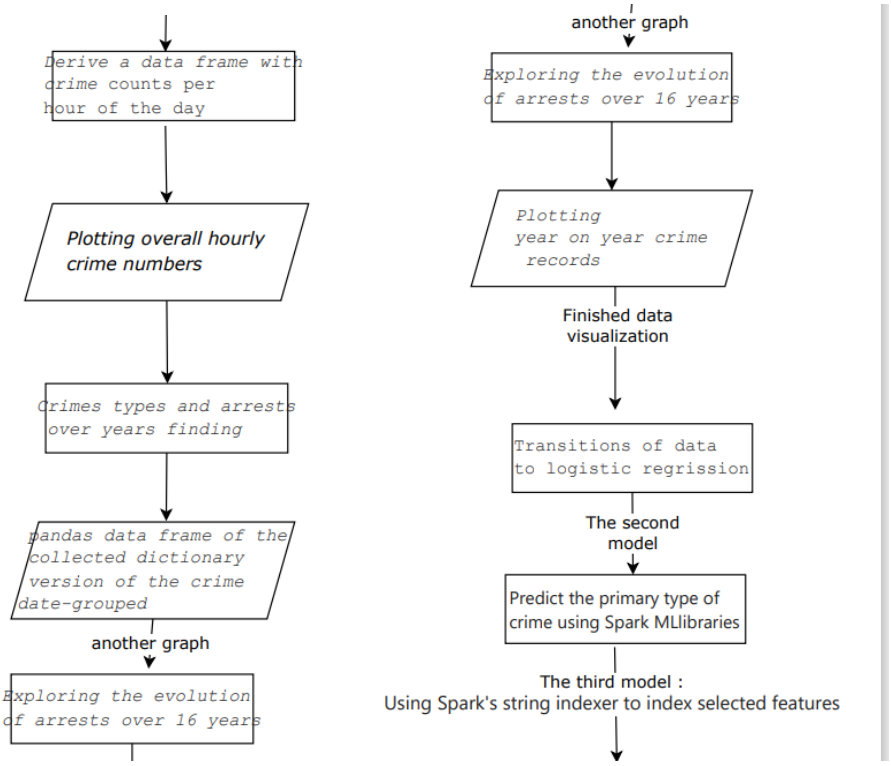


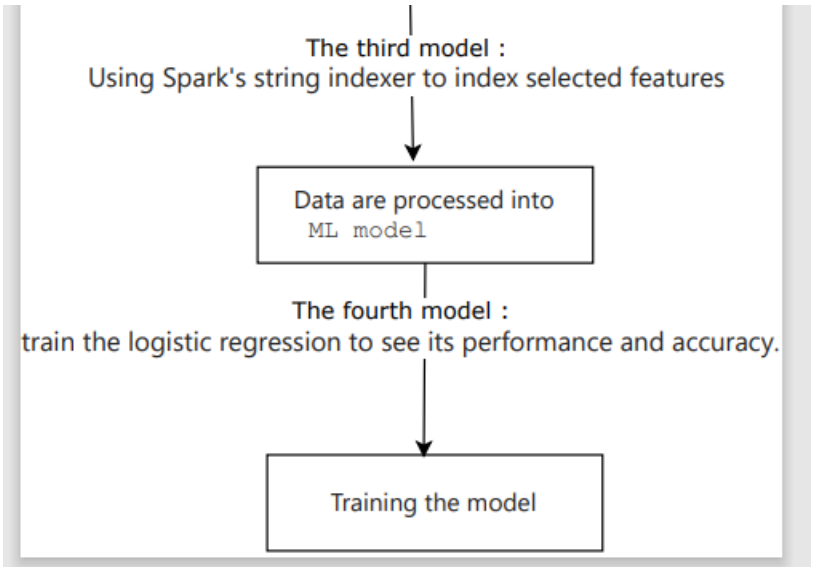
Data Flow Diagram (DFD)



Flow Charts




Derive a data frame with crime counts per hour of the day
Plotting overall hourly crime numbers
Crimes types and arrests over years finding
pandas data frame of the collected dictionary version of the crime date-grouped
another graph
Exploring the evolution of arrests over 16 years
another graph
Exploring the evolution of arrests over 16 years
Plotting year on year crime records
Finished data visualization
Transitions of data to logistic regression
The second model
Predict the primary type of crime using Spark MLlibraries
The third model :
Using Spark's string indexer to index selected features


The third model :
Using Spark's string indexer to index selected features
Data are processed into ML model
The fourth model :
train the logistic regression to see its performance and accuracy.
Training the model

Implementation

About Dataset

About Dataset (on Kaggle, here: https://www.kaggle.com/djonafegnem/chicago-crime-data-analysis). Context: This dataset reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to 2017. Data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. In order to protect the privacy of crime victims, addresses are shown at the block level only and specific locations are not identified. Should you have questions about this dataset, you may contact the Research & Development Division of the Chicago Police Department at 312.745.6071 or RDAnalysis@chicagopolice.org. Disclaimer: These crimes may be based upon preliminary information supplied to the Police Department by the reporting parties that have not been verified. The preliminary crime classifications may be changed at a later date based upon additional investigation and there is always the possibility of mechanical or human error. Therefore, the Chicago Police Department does not guarantee (either expressed or implied) the accuracy, completeness, timeliness, or correct sequencing of the information and the information should not be used for comparison purposes over time. The Chicago Police Department will not be responsible for any error or omission, or for the use of, or the results obtained from the use of this information. All data visualizations on maps should be considered approximate and attempts to derive specific addresses are strictly prohibited. The Chicago Police Department is not responsible for the content of any off-site pages that are referenced by or that reference this web page other than an official City of Chicago or Chicago Police Department web page. The user specifically acknowledges that the Chicago Police Department is not responsible for any defamatory, offensive, misleading, or illegal conduct of other users, links, or third parties and that the risk of injury from the foregoing rests entirely with the user. The unauthorized use of the words "Chicago Police Department," "Chicago Police," or any colorable imitation of these words or the unauthorized use of the Chicago Police Department logo is unlawful. This web page does not, in any way, authorize such use. Data are updated daily. The dataset contains more than 6,000,000 records/rows of data and cannot be viewed in full in Microsoft Excel. To access a list of Chicago Police Department - Illinois Uniform Crime Reporting (IUCR) codes, go to http://data.cityofchicago.org/Public-Safety/Chicago-Police-Department-Illinois-Uniform-Crime-R/c7ck-438e.

Features Description

ID - Unique identifier for the record.

Case Number - The Chicago Police Department RD Number (Records Division Number), which is unique to the incident.

Date - Date when the incident occurred. this is sometimes a best estimate.

Block - The partially redacted address where the incident occurred, placing it on the same block as the actual address.

IUCR - The Illinois Unifrom Crime Reporting code. This is directly linked to the Primary Type and Description. See the list of IUCR codes at https://data.cityofchicago.org/d/c7ck-438e.

Primary Type - The primary description of the IUCR code.

Description - The secondary description of the IUCR code, a subcategory of the primary description.

Location Description - Description of the location where the incident occurred.

Arrest - Indicates whether an arrest was made.

Domestic - Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act.

Beat - Indicates the beat where the incident occurred. A beat is the smallest police geographic **area –** each beat has a dedicated police beat car. Three to five beats make up a police sector, and three sectors make up a police district. The Chicago Police Department has 22 police districts. See the beats at https://data.cityofchicago.org/d/aerh-rz74.

District - Indicates the police district where the incident occurred. See the districts at https://data.cityofchicago.org/d/fthy-xz3r.

Ward - The ward (City Council district) where the incident occurred. See the wards at https://data.cityofchicago.org/d/sp34-6z76.

Community Area - Indicates the community area where the incident occurred. Chicago has 77 community areas. See the community areas at https://data.cityofchicago.org/d/cauq-8yn6.

FBI Code - Indicates the crime classification as outlined in the FBI's National Incident-Based Reporting System (NIBRS). See the Chicago Police Department listing of these classifications at http://gis.chicagopolice.org/clearmap_crime_sums/crime_types.html.

X Coordinate - The x coordinate of the location where the incident occurred in State Plane Illinois East NAD 1983 projection. This location is shifted from the actual location for partial redaction but falls on the same block.

Y Coordinate - The y coordinate of the location where the incident occurred in State Plane Illinois East NAD 1983 projection. This location is shifted from the actual location for partial redaction but falls on the same block.

Year - Year the incident occurred.

Updated On - Date and time the record was last updated.

Latitude - The latitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.

Longitude - The longitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block.

Location - The location where the incident occurred in a format that allows for creation of maps and other geographic operations on this data portal.

Code Snippets

Reading the data

Using the Spark's csv reader to parse the files. It processes multiple files and returns a single data frame:

```
spark = SparkSession.builder.appName("Crime").getOrCreate()
```

```
df0 = spark.read.csv('*.csv', inferSchema=True, header=True)
```

Data schema

```
df0.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- 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: string (nullable = true)
 |-- Domestic: string (nullable = true)
 |-- Beat: string (nullable = true)
 |-- District: string (nullable = true)
 |-- Ward: string (nullable = true)
 |-- Community Area: string (nullable = true)
 |-- FBI Code: string (nullable = true)
 |-- X Coordinate: string (nullable = true)
 |-- Y Coordinate: string (nullable = true)
```

Taking an initial look at the content of the data frame

```
df.show(n=3, truncate=False)
```

```
+----+----+-----------+---------------------+---------------+----+------------+-------------------+--------------------+------
+--------+----+--------+----+--------------+--------+------------+------------+----+---------------------+------------+-------
------+-----------------------------+
|_c0 |id  |case_number|date                 |block          |iucr|primary_type|description        |location_description|arrest
|domestic|beat|district|ward|community_area|fbi_code|x_coordinate|y_coordinate|year|updated_on           |latitude    |longitu
de    |location                     |
+----+----+-----------+---------------------+---------------+----+------------+-------------------+--------------------+------
+--------+----+--------+----+--------------+--------+------------+------------+----+---------------------+------------+-------
------+-----------------------------+
|388 |4785|HP610824   |10/07/2008 12:39:00 PM|000XX E 75TH ST|0110|HOMICIDE    |FIRST DEGREE MURDER|ALLEY               |True
|False   |323 |3.0     |6.0 |69.0          |01A     |1178207.0   |1855308.0   |2008|08/17/2015 03:03:40 PM|41.758275857|-87.622
451031|(41.758275857, -87.622451031)|
|835 |4786|HP616595   |10/09/2008 03:30:00 AM|048XX W POLK ST|0110|HOMICIDE    |FIRST DEGREE MURDER|STREET              |True
|False   |1533|15.0    |24.0|25.0          |01A     |1144200.0   |1895857.0   |2008|08/17/2015 03:03:40 PM|41.87025207 |-87.746
069362|(41.87025207, -87.746069362) |
|1334|4787|HP616904   |10/09/2008 08:35:00 AM|030XX W MANN DR|0110|HOMICIDE    |FIRST DEGREE MURDER|PARK PROPERTY       |False
|False   |831 |8.0     |18.0|66.0          |01A     |1157314.0   |1859778.0   |2008|08/17/2015 03:03:40 PM|41.770990476|-87.698
901469|(41.770990476, -87.698901469)|
+----+----+-----------+---------------------+---------------+----+------------+-------------------+--------------------+------
+--------+----+--------+----+--------------+--------+------------+------------+----+---------------------+------------+-------
------+-----------------------------+
only showing top 3 rows
```

How many features do we have

```
['ID',
 'Case Number',
 'Date',
 'Block',
 'IUCR',
 'Primary Type',
 'Description',
 'Location Description',
 'Arrest',
 'Domestic',
 'Beat',
 'District',
 'Ward',
 'Community Area',
 'FBI Code',
 'X Coordinate',
 'Y Coordinate',
 'Year',
 'Updated On',
 'Latitude',
 'Longitude',
 'Location']
```

Crime types

These are the top 20 most frequent crime types:

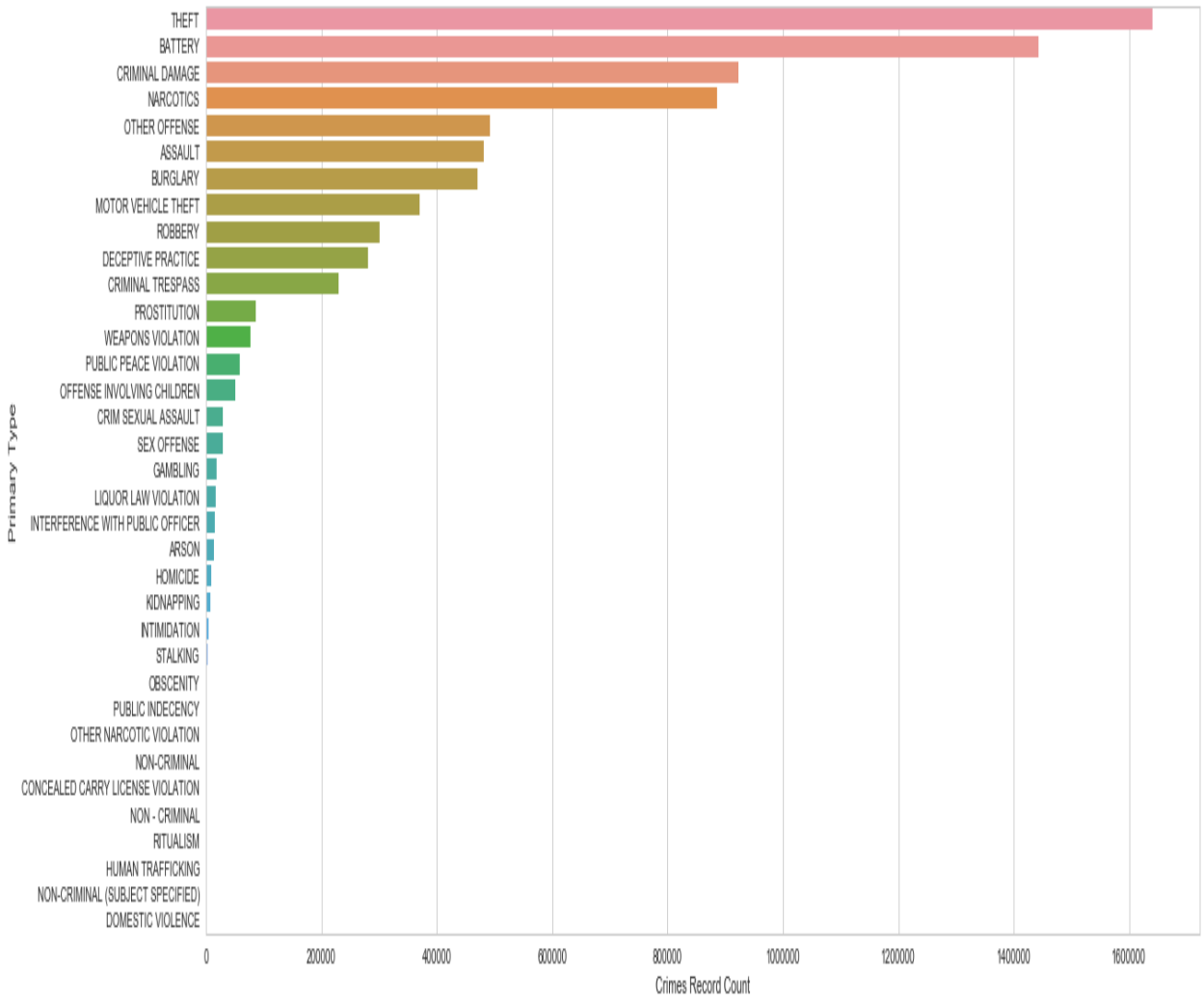
```
crime_type_counts.show(truncate=False)
```

```
+--------------------------------+-------+
|primary_type                    |count  |
+--------------------------------+-------+
|THEFT                           |1640506|
|BATTERY                         |1442717|
|CRIMINAL DAMAGE                 |923000 |
|NARCOTICS                       |885431 |
|OTHER OFFENSE                   |491923 |
|ASSAULT                         |481661 |
|BURGLARY                        |470958 |
|MOTOR VEHICLE THEFT             |370548 |
|ROBBERY                         |300453 |
|DECEPTIVE PRACTICE              |280931 |
|CRIMINAL TRESPASS               |229367 |
|PROSTITUTION                    |86401  |
|WEAPONS VIOLATION               |77429  |
|PUBLIC PEACE VIOLATION          |58548  |
|OFFENSE INVOLVING CHILDREN      |51441  |
|CRIM SEXUAL ASSAULT             |29868  |
|SEX OFFENSE                     |28707  |
|GAMBLING                        |18806  |
|LIQUOR LAW VIOLATION            |17513  |
|INTERFERENCE WITH PUBLIC OFFICER|15710  |
+--------------------------------+-------+
only showing top 20 rows
```

Visualization

```
sns.set(style="whitegrid")
sns.set_color_codes("pastel")

#sns.despine(left=True, bottom=True)
type_graph = sns.barplot(x='count', y='primary_type', data=counts_pddf)
type_graph.set(ylabel="Primary Type", xlabel="Crimes Record Count")
```



Recorded Date

It seems that the dataset we're dealing with comprises records from 2001-01-01 to 2016-12-31

```
df.select(min('date').alias('first_record_date'), max('date').alias('latest_record_date')).show(truncate=False)
```

```
+----------------------+----------------------+
|first_record_date     |latest_record_date    |
+----------------------+----------------------+
|01/01/2001 01:00:00 AM|12/31/2016 12:56:00 AM|
+----------------------+----------------------+
```

Converting dates to a timestamp type. As seen in the schema output above, the `date` field is of `string` type, which won't be very helpful

The format specifier that seems valid for date like '02/23/2006 09:06:22 PM' is `'MM/dd/yyyy hh:mm:ss a'`

```
df = df.withColumn('date_time', to_timestamp('date', 'MM/dd/yyyy hh:mm:ss a'))\
       .withColumn('month', trunc('date_time', 'YYYY')) #adding a month column to be able to view stats on a monthly
```

```
df.select(['date','date_time', 'month'])\
  .show(n=2, truncate=False)
```

```
+----------------------+-------------------+----------+
|date                  |date_time          |month     |
+----------------------+-------------------+----------+
|10/07/2008 12:39:00 PM|2008-10-07 12:39:00|2008-01-01|
|10/09/2008 03:30:00 AM|2008-10-09 03:30:00|2008-01-01|
+----------------------+-------------------+----------+
only showing top 2 rows
```

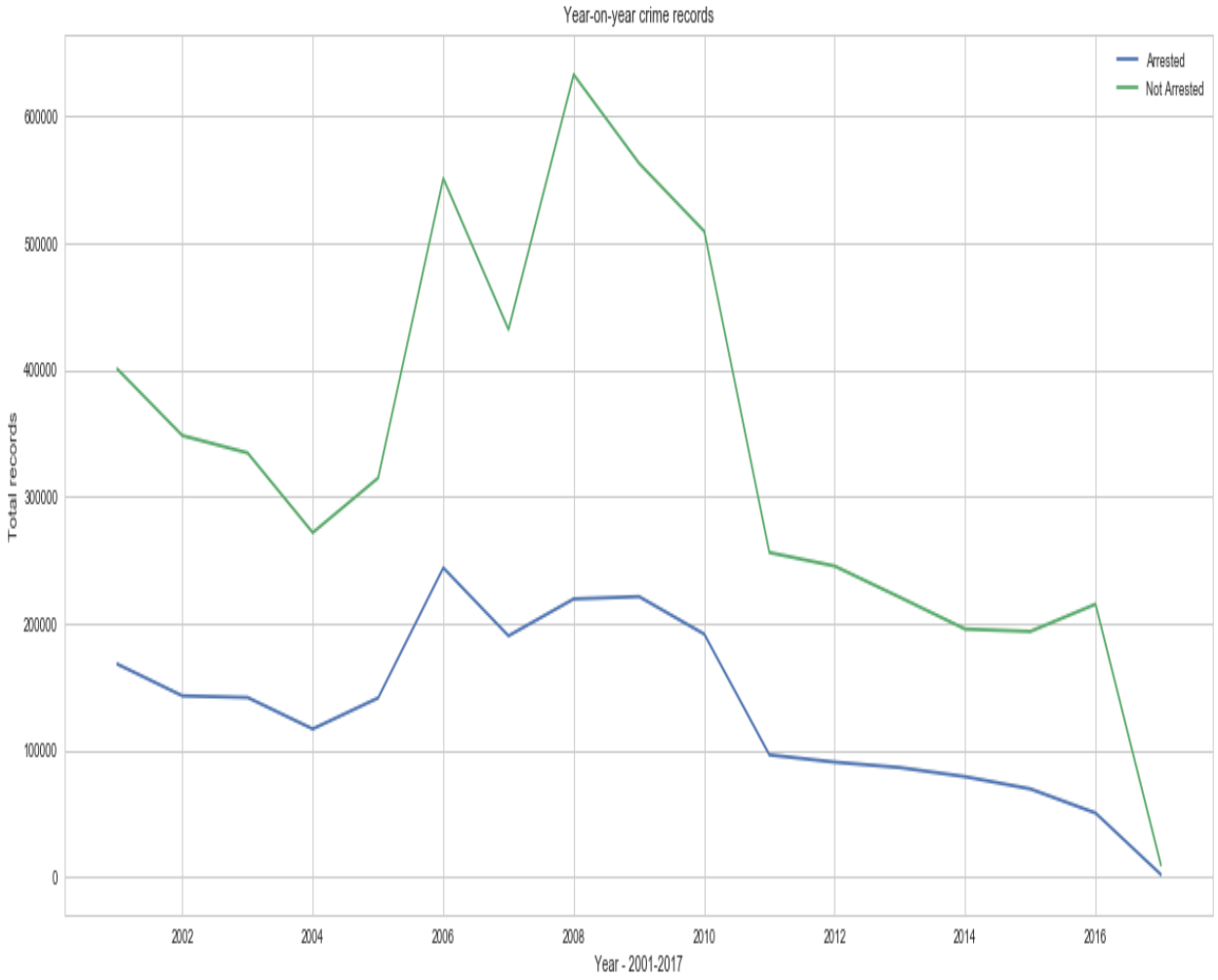
Primary type and arrest

```
# crime types and arrest over years:
type_arrest_date = df.groupBy(['arrest', 'month'])\
                     .count()\
                     .orderBy(['month', 'count'], ascending=[True, False])
print()
type_arrest_date.show(3, truncate=False)
```

```
+------+----------+------+
|arrest|month     |count |
+------+----------+------+
|False |2001-01-01|400628|
|True  |2001-01-01|167890|
|False |2002-01-01|348074|
+------+----------+------+
only showing top 3 rows
```

How have arrests evolved over the 16 years?

It looks like the relative distance between arrests and non-arrests has remained constant.



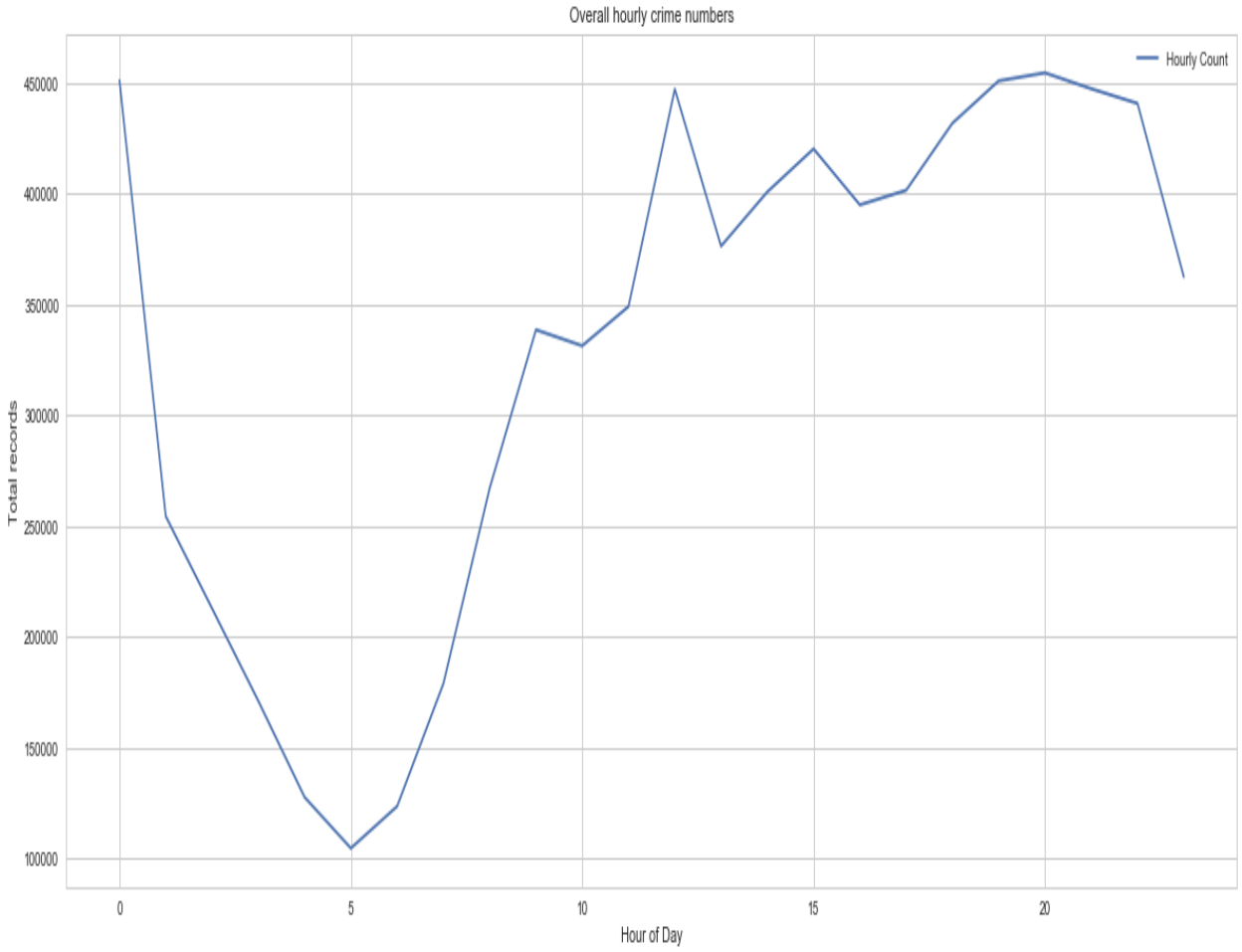
What time of the day are criminal the busiest?

Seems that 18-22 are the worst hours...

```
# Extract the "hour" field from the date into a separate column called "hour"
df_hour = df.withColumn('hour', hour(df['date_time']))
```

```
hourly_count = df_hour.groupBy(['primary_type', 'hour']).count().cache()
hourly_total_count = hourly_count.groupBy('hour').sum('count')
```

```
hourly_count_pddf = pd.DataFrame(hourly_total_count.select(hourly_total_count['hour'], hourly_total_count['sum(count)'].alias
                                 .rdd.map(lambda l: l.asDict())\
                                  .collect())
```

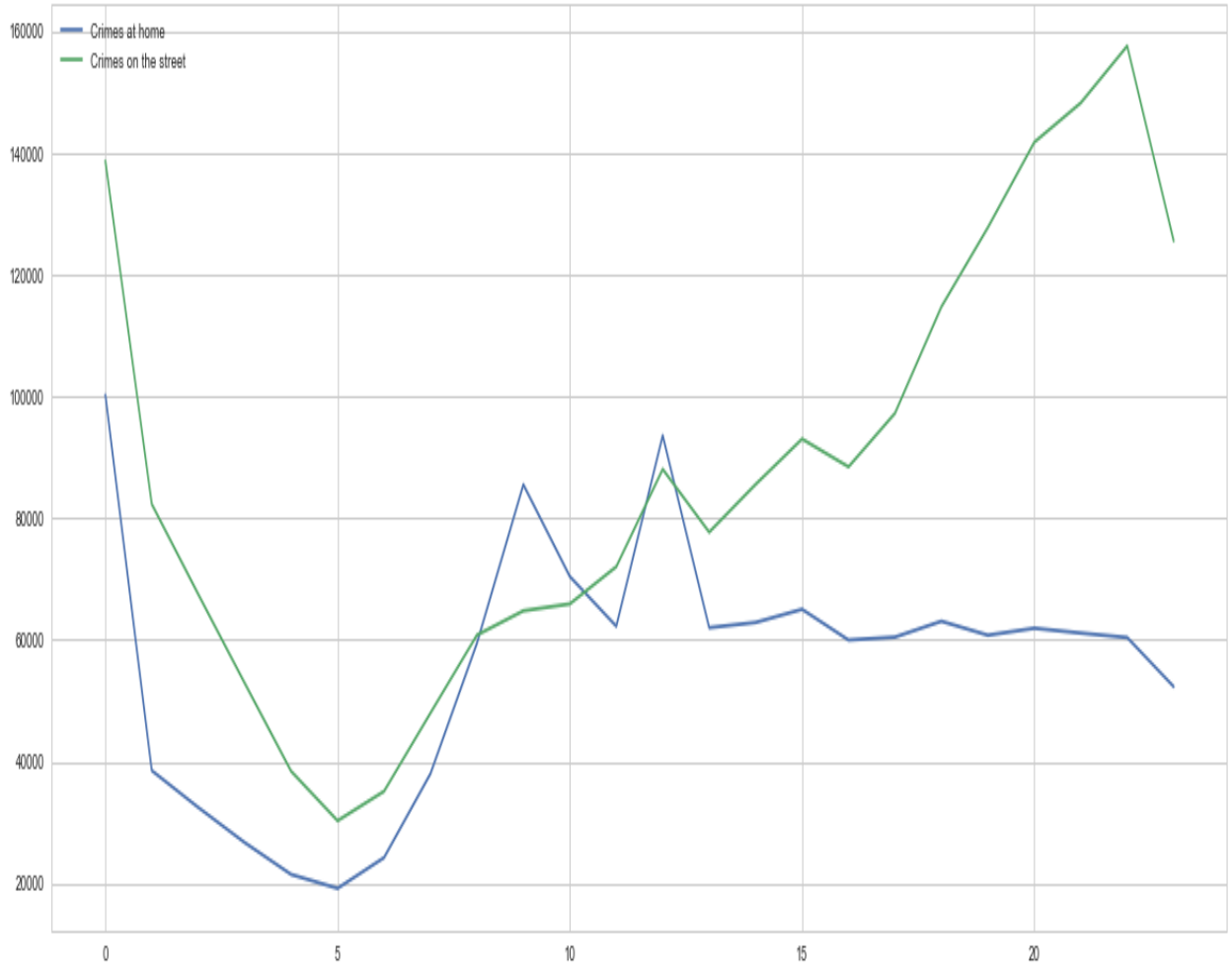


Types of locations with crime entries

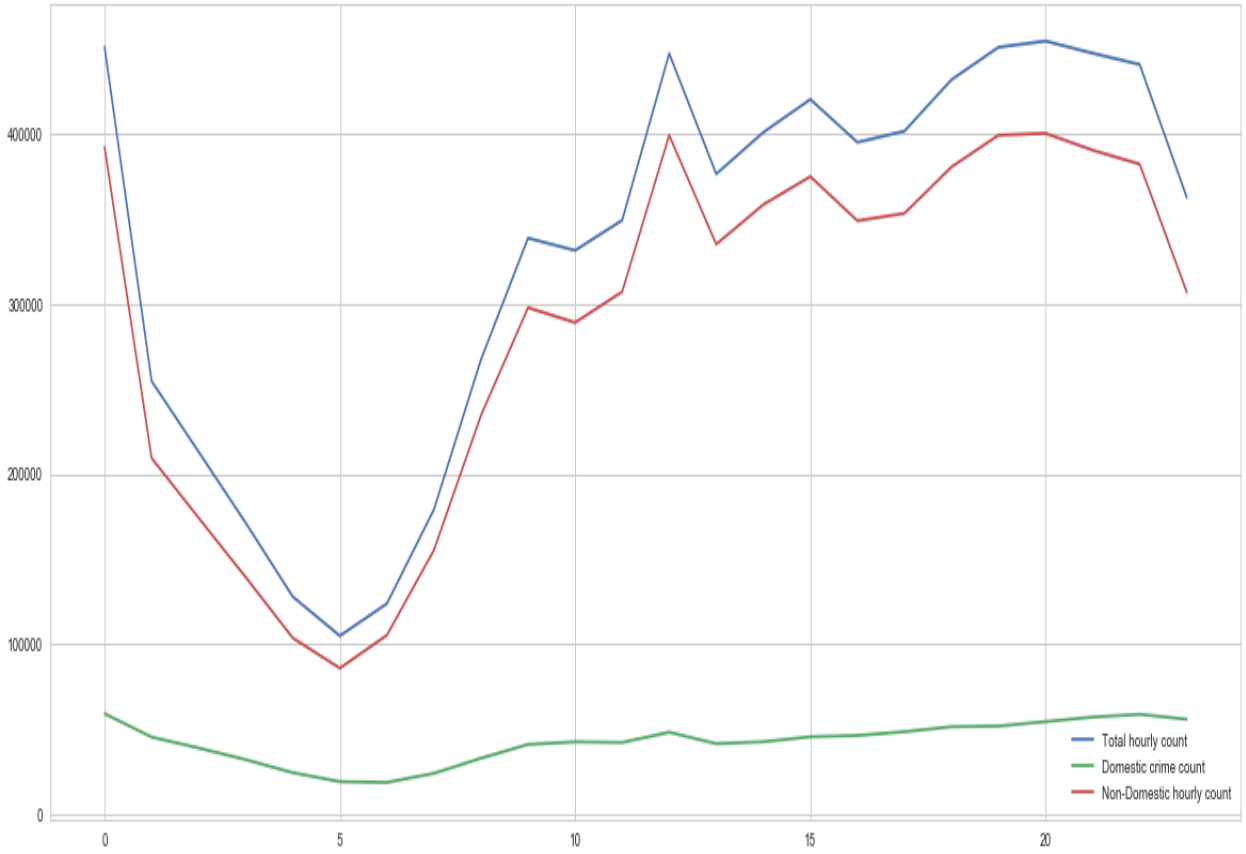
What are the top 10 places where crime occurred?

```
df.groupBy(['location_description']).count().orderBy('count', ascending=False).show(10)
```

```
+--------------------+-------+
|location_description|  count|
+--------------------+-------+
|              STREET|2101843|
|           RESIDENCE|1341750|
|            SIDEWALK| 815595|
|           APARTMENT| 812512|
|               OTHER| 294286|
|PARKING LOT/GARAG...| 225454|
|               ALLEY| 180155|
|SCHOOL, PUBLIC, B...| 173750|
|    RESIDENCE-GARAGE| 158550|
|RESIDENCE PORCH/H...| 138492|
+--------------------+-------+
only showing top 10 rows
```


Crimes at home
Crimes on the street
160000
140000
120000
100000
80000
60000
40000
20000
0
5
10
15
20

How do domestic crimes compare the other crimes?



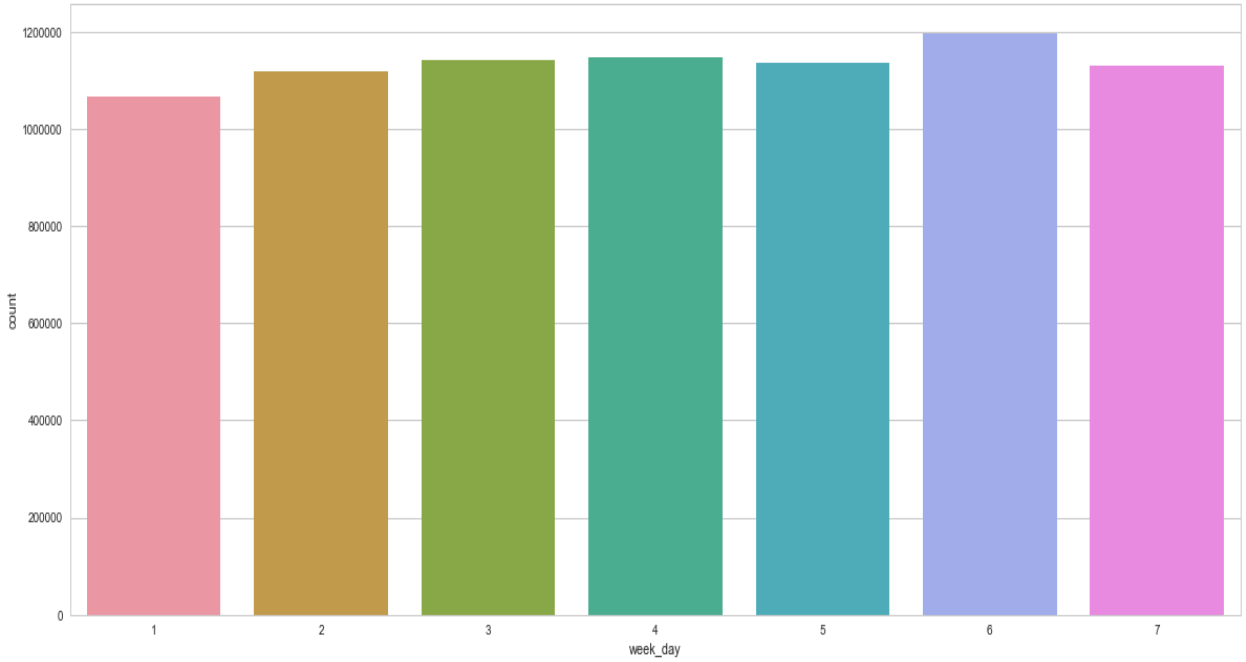
A closer look at crime date and time

```
df_dates.select(['date', 'month', 'hour', 'week_day', 'year', 'year_month', 'month_day', 'date_number']).show(20, truncate=Fa
```

```
+--------------------+----------+----+--------+----+----------+---------+-----------+
|date                |month     |hour|week_day|year|year_month|month_day|date_number|
+--------------------+----------+----+--------+----+----------+---------+-----------+
|10/07/2008 12:39:00 PM|2008-01-01|12  |3       |2008|10        |7        |2836       |
|10/09/2008 03:30:00 AM|2008-01-01|3   |5       |2008|10        |9        |2838       |
|10/09/2008 08:35:00 AM|2008-01-01|8   |5       |2008|10        |9        |2838       |
|10/10/2008 02:33:00 AM|2008-01-01|2   |6       |2008|10        |10       |2839       |
|10/10/2008 12:50:00 PM|2008-01-01|12  |6       |2008|10        |10       |2839       |
|10/10/2008 08:32:00 PM|2008-01-01|20  |6       |2008|10        |10       |2839       |
|10/11/2008 12:55:00 AM|2008-01-01|0   |7       |2008|10        |11       |2840       |
|10/11/2008 10:25:00 PM|2008-01-01|22  |7       |2008|10        |11       |2840       |
|10/11/2008 10:00:00 PM|2008-01-01|22  |7       |2008|10        |11       |2840       |
|10/12/2008 05:47:00 AM|2008-01-01|5   |1       |2008|10        |12       |2841       |
|10/12/2008 10:33:00 PM|2008-01-01|22  |1       |2008|10        |12       |2841       |
|10/12/2008 10:55:00 PM|2008-01-01|22  |1       |2008|10        |12       |2841       |
|10/13/2008 02:49:00 PM|2008-01-01|14  |2       |2008|10        |13       |2842       |
|10/13/2008 07:04:00 PM|2008-01-01|19  |2       |2008|10        |13       |2842       |
|10/14/2008 09:37:00 AM|2008-01-01|9   |3       |2008|10        |14       |2843       |
|10/14/2008 01:27:00 PM|2008-01-01|13  |3       |2008|10        |14       |2843       |
|10/15/2008 04:10:00 PM|2008-01-01|16  |4       |2008|10        |15       |2844       |
|10/16/2008 09:35:00 PM|2008-01-01|21  |5       |2008|10        |16       |2845       |
|10/16/2008 04:25:00 PM|2008-01-01|16  |5       |2008|10        |16       |2845       |
|10/17/2008 05:17:00 AM|2008-01-01|5   |6       |2008|10        |17       |2846       |
+--------------------+----------+----+--------+----+----------+---------+-----------+
only showing top 20 rows
```

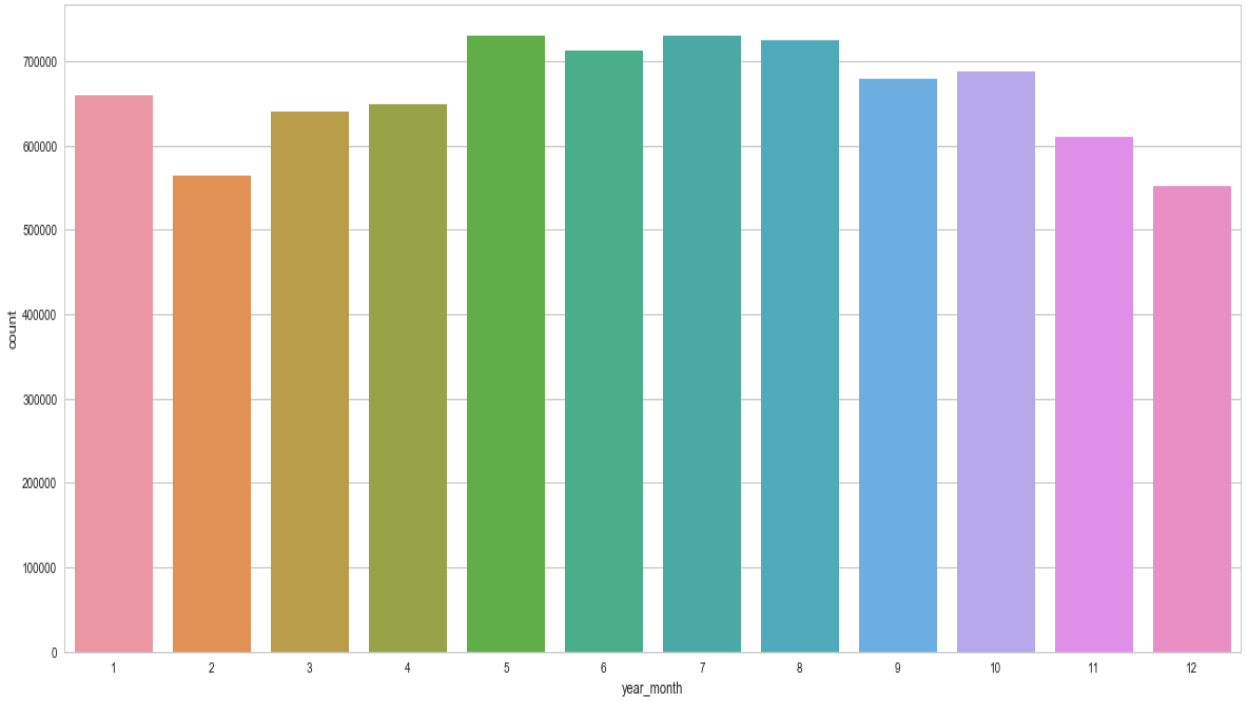
Day of week crime levels

It seems intriguing that they’re little variance.



Month of year

It seems that May-August are the busiest months for criminals.

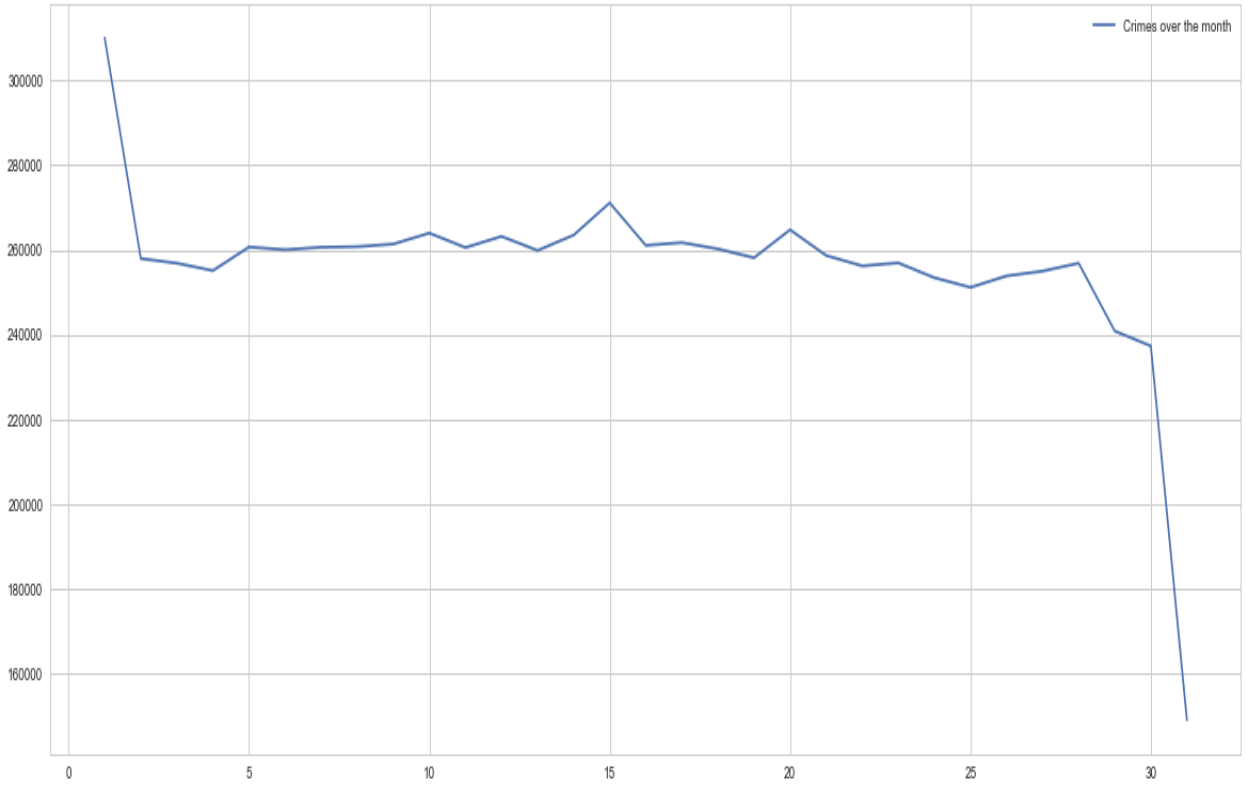


Day of month

Top 10 worst days of the month

```
month_day_crime_counts_pddf.sort_values(by='count', ascending=False).head(10)
```

	count	month_day
0	309994	1
14	271113	15
19	264762	20
9	263976	10
13	263507	14
11	263193	12
16	261727	17
8	261387	9
15	261078	16
7	260775	8



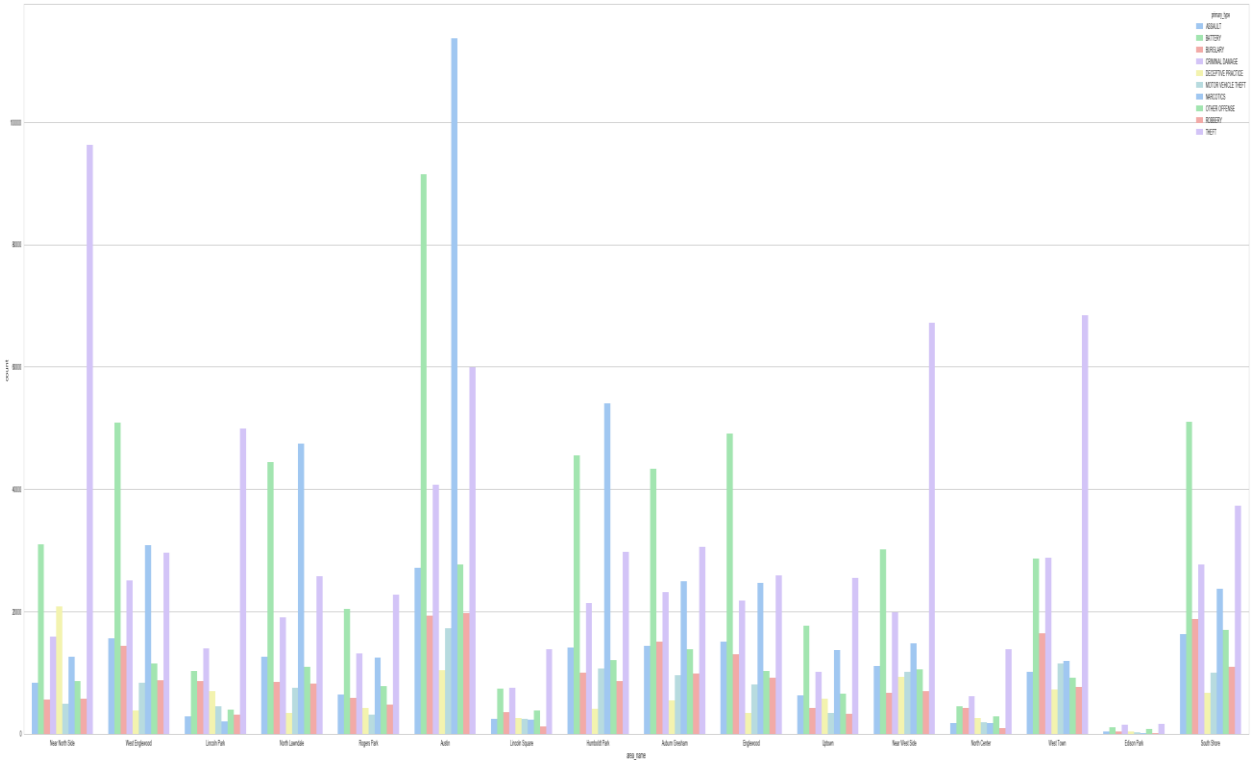
Community areas

Chicago has 77 community areas.

What are the to 10 areas with recorded crime?

```
df_dates_community_areas.orderBy('count', ascending=False).show(10)
```

```
+--------------+------+
|community_area| count|
+--------------+------+
|          25.0|467137|
|          43.0|237381|
|           8.0|235562|
|          23.0|229024|
|          67.0|216991|
|          24.0|213915|
|          71.0|207183|
|          28.0|206017|
|          29.0|202887|
|          68.0|197659|
+--------------+------+
only showing top 10 rows
```



Data postprocessing

Excluded variables:

- 'id' - Random information that isn't a predictor of crime type
- 'case_number' - Random information that isn't a predictor of crime type
- 'date' - Removed because it's been re-featurized in other features generated above
- 'block' - Excluded as this may simply mean noise
- 'iucr' - Excluded as correlated with crime type. No point.
- 'x_coordinate' - Not included
- 'y_coordinate' - Not included
- 'year' - Not included (already otherwise featurized)
- 'updated_on' - not included
- 'latitude' - not included
- 'longitude' - not included
- 'location' - not included
- 'date_time' - Taken into account in other time-related features
- 'description' - Excluded. I want to see this as associated with the response (primary type)

Selected features:

- 'location_description'
- 'arrest'
- 'domestic'
- 'beat'
- 'district'
- 'ward'
- 'community_area'
- 'fbi_code'
- 'hour'
- 'week_day'
- 'year_month'
- 'month_day'
- 'date_number'

```
selected_features = [
 'location_description',
 'arrest',
 'domestic',
 'beat',
 'district',
 'ward',
 'community_area',
 'fbi_code',
 'hour',
 'week_day',
 'year_month',
 'month_day',
```

```
features_df = df_dates.select(selected_features)
features_df.printSchema()
```

```
root
 |-- location_description: string (nullable = true)
 |-- arrest: string (nullable = true)
 |-- domestic: string (nullable = true)
 |-- beat: string (nullable = true)
 |-- district: string (nullable = true)
 |-- ward: string (nullable = true)
 |-- community_area: string (nullable = true)
 |-- fbi_code: string (nullable = true)
 |-- hour: integer (nullable = true)
 |-- week_day: integer (nullable = true)
 |-- year_month: integer (nullable = true)
 |-- month_day: integer (nullable = true)
 |-- date_number: integer (nullable = true)
```

Preparing model

```
from pyspark.ml.feature import StringIndexer, VectorAssembler
df_dates_features = df_dates.na.drop(subset=selected_features)
```

Let us use Spark's string indexer to index selected features

```
for feature in feature_level_count_dic:
    indexer = StringIndexer(inputCol=feature['feature'], outputCol='%s_indexed' % feature['feature'])
    print('Fitting feature "%s"' % feature['feature'])
    model = indexer.fit(df_dates_features)
    print('Transforming "%s"' % feature['feature'])
```

```
#what does it look like now...
df_dates_features.show(1)
```

```
+---+----+-----------+--------------------+--------------+----+------------+--------------------+--------------------+------+--
------+----+--------+----+--------------+--------+------------+------------+----+--------------------+------------+-----------
-+--------------------+------------------+----------+----+--------+----------+---------+-----------+-----------------------
--+--------------+----------------+------------+---------------+-----------+---------------------+----------------+---------
---+--------------+-----------------+--------------------+---------------+--------------------+
|_c0|  id|case_number|                date|          block|iucr|primary_type|        description|location_description|arrest|do
mestic|beat|district|ward|community_area|fbi_code|x_coordinate|y_coordinate|year|          updated_on|    latitude|    longitud
e|            location|          date_time|     month|hour|week_day|year_month|month_day|date_number|location_description_index
ed|arrest_indexed|domestic_indexed|beat_indexed|district_indexed|ward_indexed|community_area_indexed|fbi_code_indexed|hour_inde
xed|week_day_indexed|year_month_indexed|month_day_indexed|date_number_indexed|primary_type_indexed|
+---+----+-----------+--------------------+--------------+----+------------+--------------------+--------------------+------+--
------+----+--------+----+--------------+--------+------------+------------+----+--------------------+------------+-----------
-+--------------------+------------------+----------+----+--------+----------+---------+-----------+-----------------------
--+--------------+----------------+------------+---------------+-----------+---------------------+----------------+---------
---+--------------+-----------------+--------------------+---------------+--------------------+
|388|4785|   HP610824|10/07/2008 12:39:...|000XX E 75TH ST|0110|    HOMICIDE|FIRST DEGREE MURDER|              ALLEY|  True|
False| 323|     3.0| 6.0|          69.0|     01A|   1178207.0|   1855308.0|2008|08/17/2015 03:03:...|41.758275857|-87.622451031
|(41.758275857, -8...|2008-10-07 12:39:00|2008-01-01|  12|       3|        10|        7|       2836|                          6.
0|           1.0|             0.0|        47.0|            6.0|         7.0|                 12.0|            22.0|
3.0|             2.0|              4.0|            12.0|              666.0|                21.0|
+---+----+-----------+--------------------+--------------+----+------------+--------------------+--------------------+------+--
------+----+--------+----+--------------+--------+------------+------------+----+--------------------+------------+-----------
-+--------------------+------------------+----------+----+--------+----------+---------+-----------+-----------------------
--+--------------+----------------+------------+---------------+-----------+---------------------+----------------+---------
---+--------------+-----------------+--------------------+---------------+--------------------+
only showing top 1 row
```

Let's vectorize the features

We use a vector assembler to vectorize all predictors into a `features` column

```
indexed_features = ['%s_indexed' % fc['feature'] for fc in feature_level_count_dic]
indexed_features
```

```
['location_description_indexed',
 'arrest_indexed',
 'domestic_indexed',
 'beat_indexed',
 'district_indexed',
 'ward_indexed',
 'community_area_indexed',
 'fbi_code_indexed',
 'hour_indexed',
 'week_day_indexed',
 'year_month_indexed',
 'month_day_indexed',
 'date_number_indexed']
```

Now the ML part

We're using 60% to 40% split between the train and the test datasets.

Logistic Regression

```
train, test = vectorized_df_dates.randomSplit([0.6, 0.4])
```

```
from pyspark.ml.classification import LogisticRegression
```

```
logisticRegression = LogisticRegression(labelCol='primary_type_indexed', featuresCol='features', maxIter=10, family='multinom
```

```
fittedModel = logisticRegression.fit(train)
```

Model performance?

Accuracy

```
fittedModel.summary.accuracy
```

```
0.6258653555581762
```

Accuracy of this model was 0.626

Coefficient Matrix

```
fittedModel.coefficientMatrix
```

```
DenseMatrix(34, 13, [0.0114, -0.6102, 0.6461, 0.0023, 0.0201, 0.0086, 0.003, -9.3577, ..., 0.0004, 0.0002, 0.0005, 0.0005, 0.00
17, 0.0019, 0.0007, 0.0], 1)
```

Why the 34X13 shape?

That's because the multinomial logistic regression is fitted on **each class** of the label. It computes the probability of each class and then predicts based on these probabilities.

```
: print(fittedModel.coefficientMatrix)
```

```
DenseMatrix([[ 1.13672599e-02, -6.10221789e-01,  6.46076073e-01,
               2.29954405e-03,  2.01354731e-02,  8.64686710e-03,
               2.98634670e-03, -9.35774221e+00,  4.28544620e-02,
               1.46585666e-01,  6.07077354e-02,  3.22073842e-02,
               1.17925812e-04],
             [ 6.53353822e-03, -5.22149156e-01,  3.08309448e+00,
               1.20173218e-03,  1.25426550e-02, -2.24432302e-03,
               5.69843222e-04, -7.33461430e-01,  4.03531093e-02,
               1.34493468e-01,  4.14426084e-02,  2.35979902e-02,
               1.73265883e-05],
             [-1.38258906e-03, -1.65483566e+00,  5.72261365e-01,
               1.15783025e-03, -5.16405923e-04,  1.50914357e-02,
               5.94493447e-03, -1.04036717e+00,  4.35902488e-02,
               1.40044651e-01,  5.10929987e-02,  2.57467987e-02,
               2.10966867e-05],
             [-3.21085079e-02,  8.17549476e+00, -9.56339371e+00,
               2.25578321e-04, -3.37413100e-03, -1.39061526e-02,
              -6.57495097e-04, -6.31895025e-01, -1.13607948e-02,
               9.76346363e-02,  6.96949733e-02,  2.40980318e-02,
```

```
print('Coefficient matrix:\nRow count = %s\nCol count = %s' % (fittedModel.coefficientMatrix.numRows, fittedModel.coefficientM
```

```
Coefficient matrix:
Row count = 34
Col count = 13
```

```
print('Model:\nNum Classes = %s\nNum Features = %s' % (fittedModel.numClasses, fittedModel.numFeatures))
```

```
Model:
Num Classes = 34
Num Features = 13
```

```
rates_pddf
```

	f_measure	false_positive_rate	index	precision_rate	primary_type	recall_rate	true_positive_rate
0	1.000000	0.000000	0.0	1.000000	THEFT	1.000000	1.000000
14	0.432843	0.000941	14.0	0.691436	OFFENSE INVOLVING CHILDREN	0.315025	0.315025
3	0.771000	0.068760	3.0	0.639363	NARCOTICS	0.970897	0.970897
6	0.694052	0.037359	6.0	0.590063	BURGLARY	0.842537	0.842537
2	0.660851	0.079295	2.0	0.567407	CRIMINAL DAMAGE	0.791139	0.791139
1	0.453564	0.083041	1.0	0.517437	BATTERY	0.403727	0.403727
7	0.535707	0.025021	7.0	0.517173	MOTOR VEHICLE THEFT	0.555620	0.555620
18	0.348696	0.000714	18.0	0.452954	LIQUOR LAW VIOLATION	0.283453	0.283453
11	0.566450	0.011726	11.0	0.431527	PROSTITUTION	0.824125	0.824125
17	0.469533	0.001852	17.0	0.416885	GAMBLING	0.537402	0.537402
4	0.045514	0.002276	4.0	0.410242	OTHER OFFENSE	0.024093	0.024093
10	0.215583	0.006262	10.0	0.410169	CRIMINAL TRESPASS	0.146217	0.146217
21	0.303626	0.000434	21.0	0.396214	HOMICIDE	0.246114	0.246114
13	0.031158	0.000194	13.0	0.389658	PUBLIC PEACE VIOLATION	0.016228	0.016228
9	0.410013	0.027827	9.0	0.376070	DECEPTIVE PRACTICE	0.450692	0.450692
5	0.360216	0.037348	5.0	0.374041	ASSAULT	0.347376	0.347376
8	0.291963	0.023245	8.0	0.315355	ROBBERY	0.271801	0.271801
12	0.264141	0.005356	12.0	0.302200	WEAPONS VIOLATION	0.234597	0.234597
20	0.279198	0.002067	20.0	0.223772	ARSON	0.371121	0.371121
15	0.022451	0.000322	15.0	0.127900	CRIM SEXUAL ASSAULT	0.012305	0.012305
16	0.000643	0.000028	16.0	0.039683	SEX OFFENSE	0.000324	0.000324
27	0.000000	0.000000	27.0	0.000000	OTHER NARCOTIC VIOLATION	0.000000	0.000000
32	0.000000	0.000000	32.0	0.000000	RITUALISM	0.000000	0.000000
31	0.000000	0.000000	31.0	0.000000	HUMAN TRAFFICKING	0.000000	0.000000
30	0.000000	0.000000	30.0	0.000000	NON - CRIMINAL	0.000000	0.000000
29	0.000000	0.000000	29.0	0.000000	CONCEALED CARRY LICENSE VIOLATION	0.000000	0.000000
28	0.000000	0.000000	28.0	0.000000	NON-CRIMINAL	0.000000	0.000000
22	0.000000	0.000000	22.0	0.000000	KIDNAPPING	0.000000	0.000000
26	0.000000	0.000000	26.0	0.000000	PUBLIC INDECENCY	0.000000	0.000000
25	0.000000	0.000000	25.0	0.000000	OBSCENITY	0.000000	0.000000
24	0.000000	0.000000	24.0	0.000000	STALKING	0.000000	0.000000
23	0.000000	0.000000	23.0	0.000000	INTIMIDATION	0.000000	0.000000
19	0.000000	0.000000	19.0	0.000000	INTERFERENCE WITH PUBLIC OFFICER	0.000000	0.000000
33	0.000000	0.000000	33.0	0.000000	NON-CRIMINAL (SUBJECT SPECIFIED)	0.000000	0.000000

Naïve Bayes Model

Naive Model

```
from pyspark.ml.classification import NaiveBayes

nb = NaiveBayes( modelType="multinomial" ,featuresCol='features'
                ,labelCol = "primary_type_indexed")

model = nb.fit(train)
```

```
[Stage 248:>                                                        (0 + 3) / 16]
23/01/03 04:52:51 WARN MemoryStore: Not enough space to cache rdd_127_0 in memory! (computed 92.6 MiB so far)
[Stage 248:==========>                                              (3 + 3) / 16]
23/01/03 04:53:06 WARN MemoryStore: Not enough space to cache rdd_127_5 in memory! (computed 28.2 MiB so far)
[Stage 248:===============>                                         (4 + 3) / 16]
23/01/03 04:53:10 WARN MemoryStore: Not enough space to cache rdd_127_6 in memory! (computed 53.4 MiB so far)
```

Accuracy of this model was 0.1

Model performance

```
predictions = model.transform(test)
```

```
print("Test set accuracy = " + str(accuracy2))
```

```
[Stage 251:=======>                                                 (2 + 3) / 16]
23/01/03 04:54:43 WARN MemoryStore: Not enough space to cache rdd_127_4 in memory! (computed 54.7 MiB so far)
[Stage 251:==========>                                              (3 + 3) / 16]
23/01/03 04:54:44 WARN MemoryStore: Not enough space to cache rdd_127_5 in memory! (computed 28.2 MiB so far)
[Stage 251:=============================================================>   (15 + 1) / 16]
Test set accuracy = 0.10052570089422351
```

KMeans Model

KMeans

```
from pyspark.ml.clustering import KMeans
```

```
kmeans = KMeans(k=34,featuresCol='features'
                )

kmeans.setSeed(1)
```

```
KMeans_38d72bdaecac
```

```
kmeans.setMaxIter(10)
```

```
KMeans_38d72bdaecac
```

```
model.getDistanceMeasure()
```

```
summary.clusterSizes
```

```
[158904,
 106795,
 118889,
 130790,
 72219,
 169963,
 130567,
 96760,
 129323,
 130173,
 137954,
 176148,
 130830,
 116279,
 128124,
 87455,
 197700,
 124034,
 128910,
 146529,
 129111,
 96127,
 114749,
 169030,
 103181,
 176386,
 124177,
 129175,
 124439,
 140811,
 83092,
 119675,
 122319,
 91609]
```

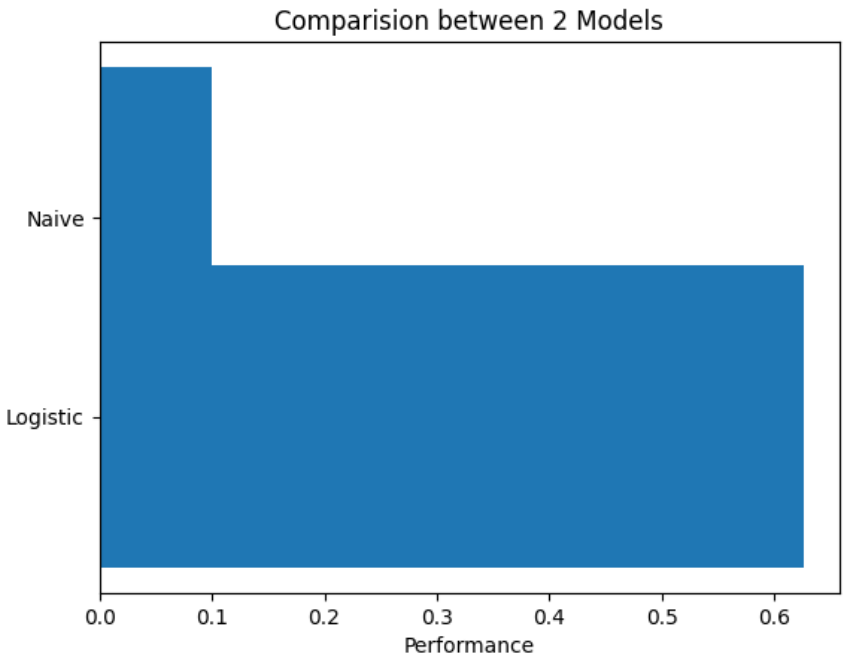
Compare the Accuracy of ML Models

The naïve bayes model is built upon independence assumptions which are often inaccurate. Where assumptions of logistic regression are: Independent variables show a linear relationship with the log of output variables, and non-Collinearity between independent variables. That is, independent variables are independent of each other Which gives a higher accuracy and can predict better than naïve model. KMeans are used to visualize the clusters which are have some similarity so we can detect the new tuple belongs to which cluster.

```
plt.rcdefaults()
fig, ax = plt.subplots()


people = ('Logistic', 'Naive')
y_pos = np.append(accuracy,accuracy2)
performance =  [accuracy,accuracy2]

ax.barh(y_pos, performance)
ax.set_yticks(y_pos, labels=people)
ax.invert_yaxis()
ax.set_xlabel('Performance')
ax.set_title('Comparision between 2 Models')
```



Conclusion

The purpose of this problem was to explore the applicability of data analysis techniques for prediction of crime types. This report provides information about crime-based incidents, showing the percentage different types of crime incidents that occur in Chicago, the areas which are more prone and sensitive to crime and the percentage of each type of crime incidents that occurs in each area. Area types considered in work are slums, residential, commercial, VIP zones, travel points and markets and crime types considered are heinous crimes, non-heinous crimes and special and local laws. It provides information that which areas are sensitive towards crime and also an association between areas and crime types. Analysis can help Chicago Police Department in analyzing crime profiles and finding potential solutions to mitigate similar incidents in future. It also helps into detect patterns and trends of crime incidents for the future purpose. Finally, an attempt has also been made to extract crime profiles hence police officials can make use of these profiles in their day-to-day battle against crime.

References

[1] Hsinchun Chen, Wingyan Chung, Yi Qin, Michael Chau, Jennifer Jie Xu, Gang Wang, Rong Zheng, Homa Atabakhsh, “Crime Data Mining: An Overview and Case Studies”, AI Lab, University of Arizona, proceedings National Conference on Digital Government Research, 2003, available at: http://ai.bpa.arizona.edu/

[2] Hsinchun Chen, Wingyan Chung, Yi Qin, Michael Chau, Jennifer Jie Xu, Gang Wang, Rong Zheng, Homa Atabakhsh, “Crime Data Mining: A General Framework and Some Examples”, IEEE Computer Society April 2004.

[3] C McCue, “Using Data Mining to Predict and Prevent Violent Crimes”, available at: http://www.spss.com/dirvideo/richmond.htm?source=dmp age&zone=rtsidebar

[4] Whitepaper, “Oracle’s Integration Hub For Justice And Public Safety”, Oracle Corp. 2004, available at: http://www.oracle.com/industries/government/Integration Hub_Justice.pdf