Final Project Group 7
Rosa Zhu (500047), Sylvia Zhang (499259), Xinya Zhang (499918), Vivian Zhong (452183)

Executive Summary

For this project, we want to use big data tools to improve Chicago's public safety. We selected a data file about crimes occurring in Chicago on Kaggle. The original source of the data is provided by Chicago Police Department. By analyzing this dataset, we want to explore the characteristics of crimes in Chicago, including time, location, severity, and potentially predict trends of future crimes and come up with plans for Chicago police and residents, to help improve public safety in Chicago.

To simplify the thinking process, we divided the problem into 5 different parts. The first two parts focus on the general characteristics of crimes happening in Chicago, including the description, time, and location of those crimes. The next two parts focus on examining the danger level and severity of those crimes. The last part attempts to make the prediction of future crimes, giving a clear guideline for the public safety sector in Chicago.

We utilized ***Hadoop*** and ***HDFS*** to upload the data file onto the server. We then utilized different tools such as ***PySpark***, ***MapReduce***, and ***Hive*** to process the data file, and get necessary data. To visualize the results, we used tools such as ***Wordclouds***, ***Tableau***, and ***DataWrapper***. We also included tools such as ***Second Exponential Smoothing*** when making predictions for future crimes.

The key finding for characteristics of crimes is that there aren't many changes across time. Since we have the data from 2001 to 2020, when we compare data from the past with recent data, we find they share the same characteristics. Most crimes are not very dangerous in nature, with "simple", "under $500" being the most usual words in the description. Across different districts, crimes are more likely to happen in summer and from noon to evening. For danger level, though, different districts have different patterns. Some districts are more dangerous than others, with higher dangerous levels and higher arrest ratios. For our final prediction and recommendation, we calculated the predicted dangerous level, and recommend strengthening police forces in District 3, 15, and 31 in the future.

There is still room for improvement for this report. For example, we could find data about Chicago police force of each district, and and set a plan targeting each districts and crime types if we have more time. However, we believe the current findings that we have are meaningful and could be used to improve Chicago public safety.

1. Description of the data

The data set we use is about Chicago crimes, containing the data on public safety in Chicago from 2001 to 2020. We found the data in Kaggle.com, and the original source is from Chicago Data Portal with the website link https://data.cityofchicago.org/Public-Safety/Crimes-2020/qzdf-xmn8. Jonathan Levy owns the dataset with data provided by Chicago Police Department, and the data set was first created on January 9, 2020 and last updated on December 2, 2021.

The data has a size of 1.68 GB with 22 columns, including ID, case number, data, 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, and location.

ID: Unique identifier for the record.
IUCR: Illinois Uniform Crime Reporting code, directly linked to the Primary Type and Description.
Primary Type: The primary description of the IUCR code.
District: The police district where the incident occurred. See the districts at https://data.cityof chicago.org/d/fthy-xz3r.
Description: Secondary description of IUCR code and the subcategory of primary description.
FBI code: Crime classification outlined in the FBI National Accident Reporting System.

2. Problem Statement

We are trying to compare crime occurrence and danger degree in different districts of Chicago, so we can gain some insights to help Chicago police deputies to strengthen security.

(1) We want to get a general overview of crimes in Chicago and the crime descriptions with high frequency.
(2) We would like to examine the time distribution of crime cases across different districts.
(3) We want to explore which crime type occurs most in each district, the distribution of different crime types and the distribution of danger level.
(4) We would like to get the arrest ratio of different police districts.
(5) We would like to observe each years' Comprehensive Danger Levels (*CDLs*) of different police districts in Chicago, thus the trend of *CDL*; and use Second Exponential Smoothing Method to predict the 2022 *CDL*, and the short-term trend for each district.

3. Why This Is Big Data

3.1. Reason To Select This Data

Public safety is one of the most critical issues for human beings. We want to gain a better understanding of the crimes in Chicago through analyzing this data set. Then we can propose a better plan for the police as well as residents to take targeted precautions to prevent more crimes in Chicago.

3.2. Why is it big data?

The data set is 1.68GB within more than 7.14m rows and 22 columns. It cannot be presented in Excel due to its large volume.

4. Methods and Results

4.1. Analysis of Crime Descriptions

4.1.1. Method (part 1)

For this part, we want to get a general overview of crimes in Chicago. So, we first put our data file into the ***HDFS*** and then used ***MapReduce*** to analyze the description column (column 7) to get the frequency of words' occurrences in crime descriptions.

4.1.2. Result (part 1)

Appendix [2.1] shows the top 20 words in descriptions. After ignoring meaningless words, we visualized the result into the pie chart below. As we can see, the word that appears most frequently is simple, and then comes $500, domestic, battery, vehicle, property, entry, automobile, cannabis, theft, forcible, 30gms, telephone.

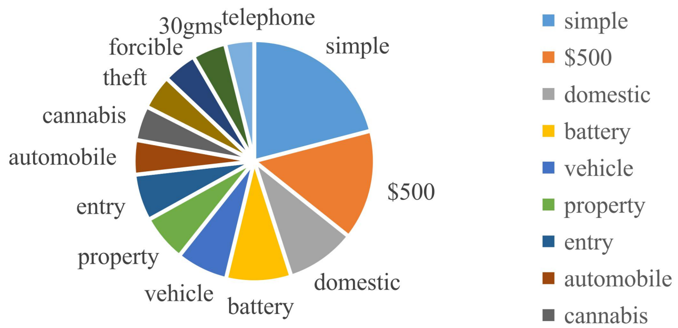

Exhibit [4.1]

4.1.3. Method (part 2)

We want to delve more deeply into this question and see if there are any changes in the top words in descriptions over time. So, we chose the year 2020, the year 2015, the year 2010, and the year 2005 to compare the most frequent crime descriptions over time.

We want to visualize the results into Wordclouds by using *wordclouds.com*. Because of the format requirements of the website for uploading files, we mainly used a mapper and then used python to change the results to the format required by the website. Appendix [2.2] shows the format required by the website, and Appendix [2.3] shows the sample results we got after changing the format using python.

4.1.4. Result (part 2)

We tried to upload the file with all the results into the website to create a wordcloud showing all the results from 2001 to 2020. However, the website cannot run a file with that much data. So, we decided to take three random samples each year to present the outcome, and selected 10000 data randomly for each sample. Below are the wordclouds for each year.

From Wordclouds, we can see that the descriptions of crimes with high frequency do not change a lot over time. Crime descriptions like simple, to vehicle, 500 and under, to property, and automobile are top descriptions for all the years.

4.2. Time Distribution Of Crime Occurrences.

For this part, we want to explore the time distribution of crime occurrences, so that we can make targeted efforts to prevent crimes.

4.2.1. Method

We mainly used Hive to extract the necessary data. Then we utilized Tableau to visualize the results from the graph.

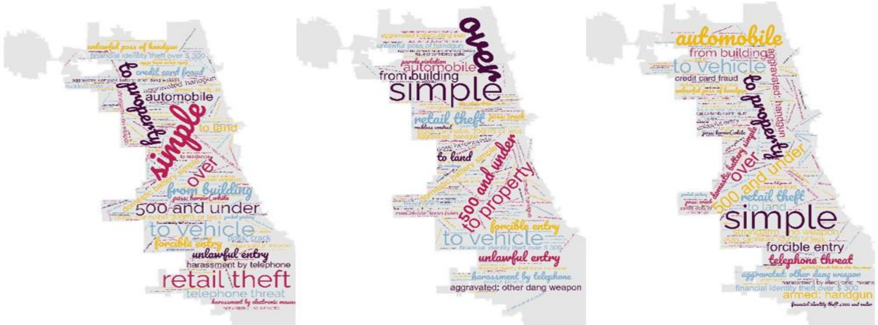

Exhibit [4.2] 2005

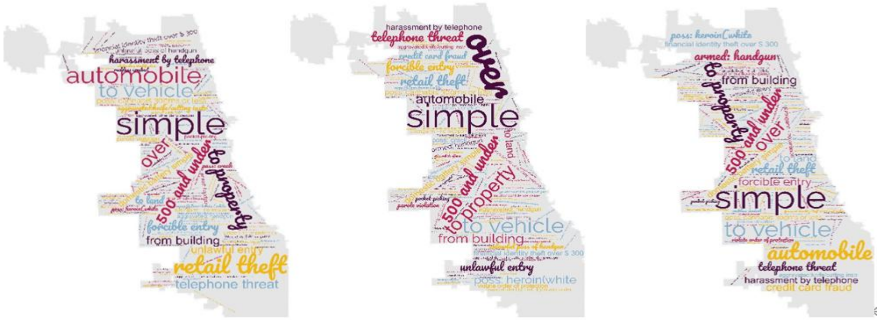

Exhibit [4.3] 2010

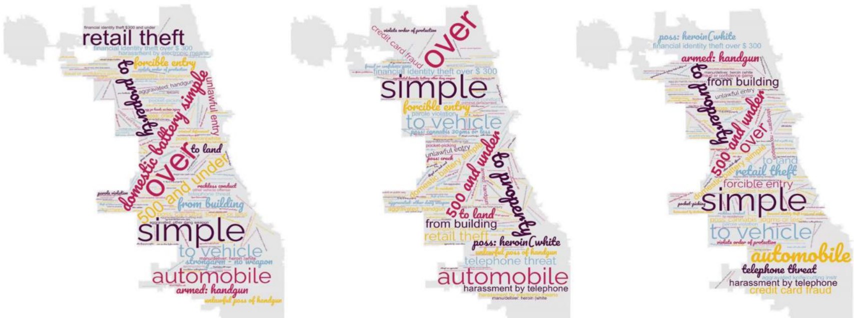

Exhibit [4.4] 2015

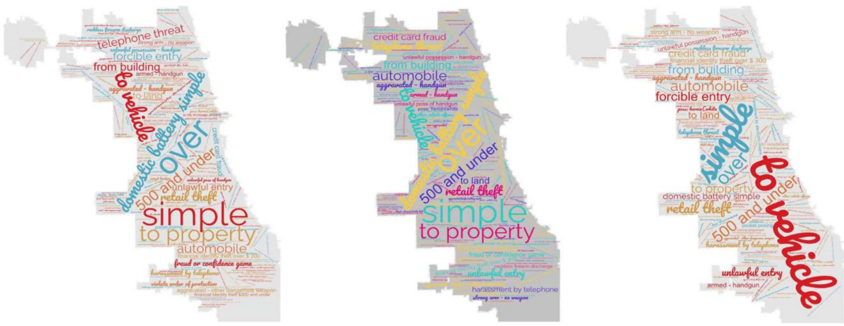

Exhibit [4.5] 2020

4.2.2. Result

We took the total number of crime occurrences of all times and grouped it by different months. We then sorted the table by crime number. We attached the table in Appendix [2.4].

With this table, we can see that colder seasons, from November to April, have fewer crimes in general. Meanwhile, July and August, usually with the highest temperature, are at the top of the table. In order to better visualize the results, we have also created a graph showing the trends using Tableau.

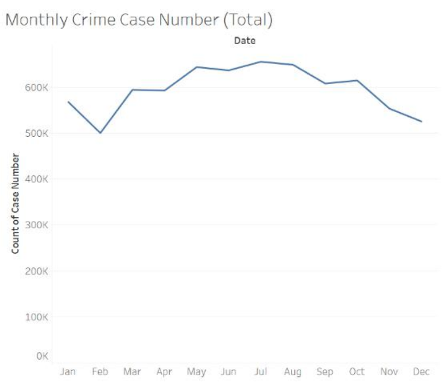

Exhibit [4.6]

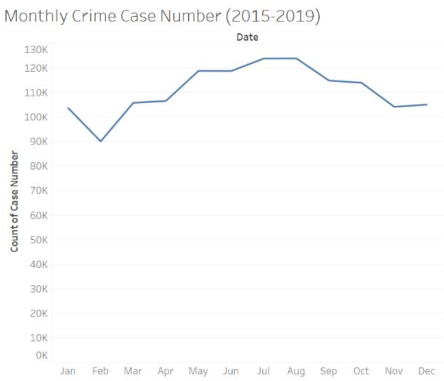

Exhibit [4.7]

From this Exhibit [4.6], we can clearly see that warmer months would have more crimes. We have also created a second graph using Tableau to filter on more recent years (2015-2019), as shown in Exhibit [4.7], and the trend stays the same.

We also want to see if this trend stays the same across all districts. Using Hive, we grouped the data by both month and district. The resulting graph is shown in Exhibit [4.8] (part of the table can be seen in Appendix [2.5]).

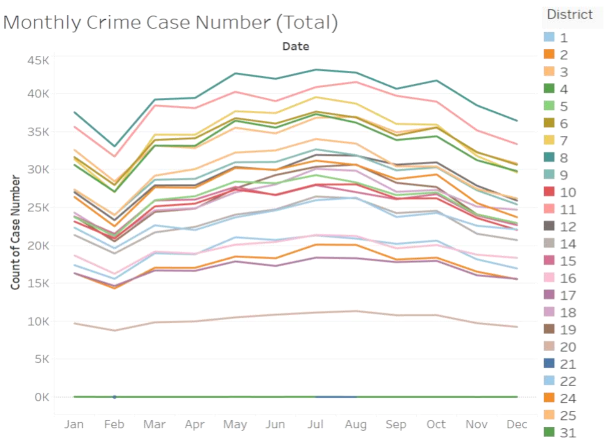

Exhibit [4.8]

As we can see from Exhibit [4.8], all districts share very similar trends in the monthly distribution of crime occurrences, with a higher occurrence rate in summer, and a lower occurrence rate in winter.

We would also like to see the hourly distribution of the crime occurrence rate. Therefore, using Hive, we got the occurrence count group by different hours using a 24-hour format. The table extracted is in Appendix [2.6] and the resulting graph that we made via Tableau are shown in Exhibit [4.9].

From and Exhibit [4.9] and Exhibit [4.10], we can see that more crimes occur from noon to midnight (12 pm-12 am), and fewer crimes occur during the early mornings. If we split the total number by district, we can see a similar pattern across different districts, too. (Although there are some minor differences for different districts. For example, district 11 has its peak at 7 pm. However, for most other districts, they have their peak at 12 pm.)

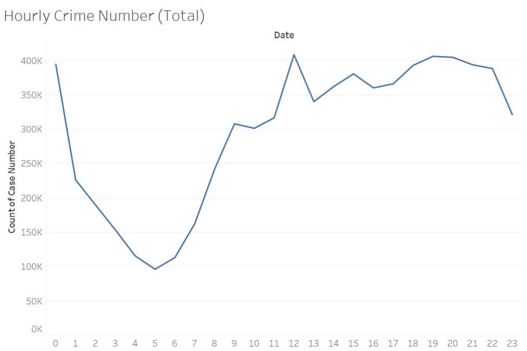

Exhibit [4.9]

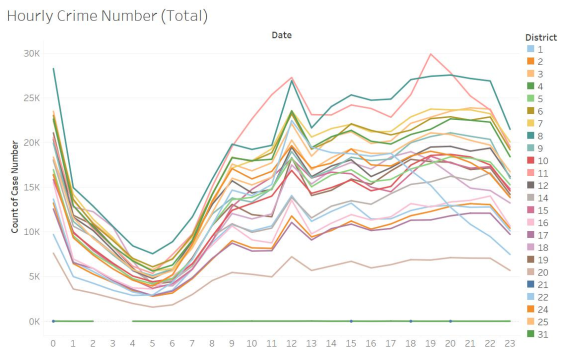

Exhibit [4.10]

With those findings, we can see that it's better to have more police officers around warmer seasons, and potentially hire more patrol officers around noons and evenings to prevent crime occurrences.

4.3. Crime Type Distribution And Its Average Danger Level

In each district, which crime type occurs most? What is the distribution of different crime types?

We have roughly divided the danger degree of the crime type into four levels. The different crime types are ranked in order of danger, with the lowest level of danger set at 1, and in ascending order of danger, the danger level +1. As shown in the table below.

Primary Type	Danger Degree		
STALKING	1	1	Non-criminal, not involving personal safety
THEFT	1	2	Involving minor personal injury
ASSAULT	2	3	Involving moderate personal injury
BURGLARY	2	4	Involved in serious personal injury resulting in death
KIDNAPPING	3		
HOMICIDE	4		

By assigning the different crime type of different danger degree, we also thought to calculate the average of Crime Type Danger Level (*CTDL*) in each district.

$$E[CTDL]_{district} = \frac{\sum (occurence\ of\ CT \cdot DL)_{district}}{(Total\ Num\ of\ Crime)_{district}} \qquad [4.1]$$

As stated in formula, this indicator can imply the dangerous level of each district in Chicago, so we can gain the general idea of the security environment in Chicago.

4.3.1. Method (part 1)

First, we used the "hdfs dfs -put" first to put our data file into the ***HDFS***, and then we used the ***Pyspark*** to divide and conquer the big data. In Spark, we created the ***RDD*** to map and transform the data, after massage the data, we grouped the data by importing the SQL queries to perform RDD count actions. In this way, we got the total number of the crime occurrence in each district of each crime type, and sorted the result in the descending order of the crime occurrence.

4.3.2. Result (part 1)

The results shows that the most frequent crime type is "Theft", "Narcotics" and "Battery", which "Theft" occurred most in District 18 "Narcotics" occurred most in District 11 and "Battery occurred most in District 7. Therefore, the local police can take some targeted precautions in different areas and strengthen the police presence in dangerous places.

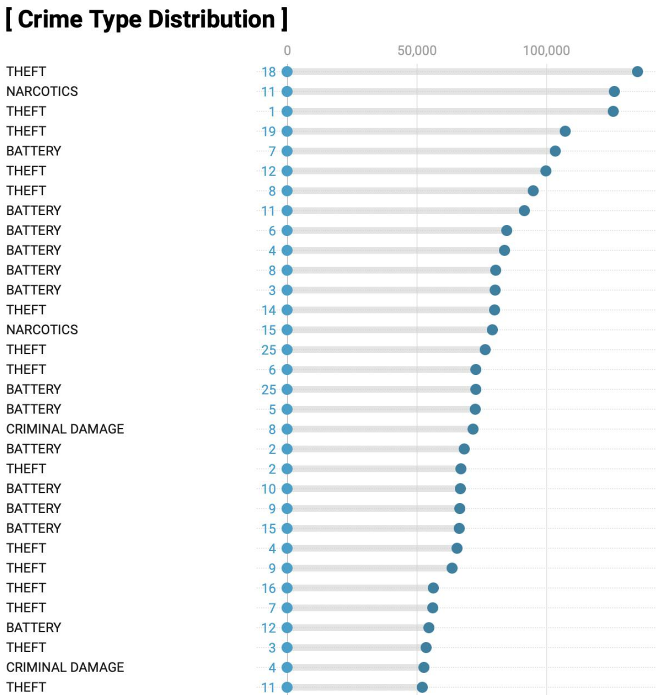

Exhibit [4.11]

4.3.3. Method (part 2)

In order to calculate the average of Crime Type Danger Level (*CTDL*) in each district, we used the ***MapReduce*** method to map a key to a value. In this case, the key is the crime type, and the value is the danger degree. We created a ***dictionary*** to assign the value to the associate key. In the Reducer python file, since the description column in this file contains many commas, we avoid this with an ***while loop statement*** to make sure the district column is in the list of districts. The detail will be explained later in the 4.5.1.(3).

4.3.4. Result (part 2)

After the MapReduce results, we can see that District 7 has the highest danger level among all the Chicago districts, and it is in line with the results of crime type distribution shown before, where District 7 has highest Battery crime occurrence. Therefore, Chicago Police Deputy should strengthen police presence in District 7, and mainly focusing on battery prevention.

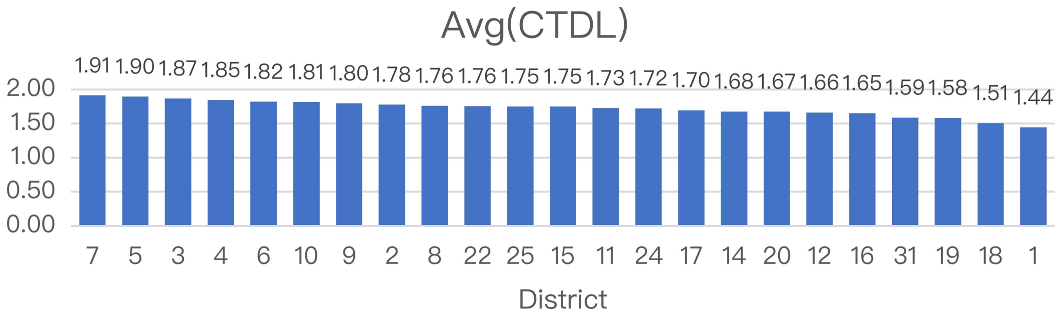

Exhibit [4.12]

4.4. Arrest Ratio Analysis

4.4.1. Method

In the fourth question, we look for the ***Arrest Ratio (AR)*** of the the 23 police districts. We define *AR* as:

$$AR = \frac{Cases\ Arrested}{Total\ Num\ of\ Cases} \qquad [4.2]$$

We used MapReduce to get the result as shown in Exhibit [4.13] (We have put the code in Appendix 1). The MapReduce is similar to the one to get the average, and for more details we will discuss in 4.5.1.(3) below.

4.4.2. Result

The result of *AR* is shown in Exhibit [4.13]. We can see the arrest ratio is highest in District 11 (*0.4322*) and 15 (*0.4094*), and is lowest in District 16 (*0.1920*).

It means District 11 and 15 in this aspect is safer than the others. Although, we can see later, the comprehensive danger level may show completely different result (for example, District 15 is high in *CDL*).

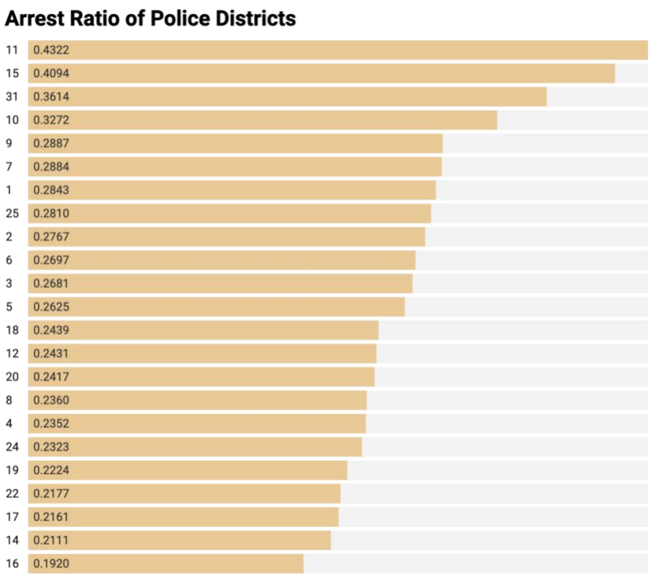

Exhibit [4.13]

4.5. Prediction of Comprehensive Danger Level

4.5.1. Method

(1) Modeling

For the fifth question, we would like to observe the trends of ***Comprehensive Danger Levels (CDLs)*** of different police districts in Chicago. To define *CDL*, we developed a simple mathematical model, as:

$$CDL = (1 - AR) \cdot \frac{1}{3} + \frac{E[CTDL]}{4} \cdot \frac{2}{3} \qquad [4.3]$$

As stated in formula [4.1], we took into consideration the ***Arrest Ratio (AR)*** and ***Crime Type Danger Level (CTDL)***.

In terms of the *AR* part, we first took the complement—*(1 - AR)* in that, the higher the *AR*, the safer the district is. It is thus reasonable to use the 'fail-to-arrest ratio'.

For the part of *CTDL*, *E[CTDL]* denotes the *arithmetic mean of CDTL* for each year, of each district. The reason why *E[CTDL]* is divided by *4* is that the highest danger level we assigned to different crime types is *4*. And in dividing by *4*, we can make the range of *E[CTDL]* be *(0,1]*, which is the same as *(1 - AR)*.

We assumed the influence of crime type is larger than the influence of arrest ratio, so we allotted more weight to *CTDL*, and less to *AR* (*1/3* and *2/3* respectively).

(2) Framework

a. Get the *AR* of each district for each year from 2001–2020.
b. Get the *E[CTDL]* of each district for each year from 2001–2020.
c. Using the model above, get the *CDL* of each district for each year from 2001–2020.
d. Based on result of step (3), use ***Second Exponential Smoothing Method*** to predict the 2022 *CDL*, and the short-term trend for each district.

(3) Big Data Processing

To complete the first two steps, we use python to create MapReduce (codes are attached in appendix).

For both MapReduce for *AR* and *E[CTDL]*, we found a common problem. That is, in the column of 'Description', comma ',' exits. Since comma is also the delimiter for csv file, and we looked for columns after the 'Description' column, it at first caused the wrong column to show up in certain rows. So we established lists and used while loops to check if the column is correct; if not, we indexed the next column, until it fit an element within the lists. Fortunately, all columns left to the column we needed did not fit any of the elements in the lists. So the method did help us to solve the problem.

Besides, for the initial index to find the correct column, we let it to be 1 digit smaller. For example, the correct column index is 11, then we let the initial index *i* to be 10 (11 - 1). This is because the 'Location' column, also left to the columns we needed, has null value, as may cause the missing of the correct column.

And in the reduce file of *E[CTDL]*, we created a dictionary *D* to get the corresponding danger level. For the values of *D*, we added '.0' to make the danger level a float, so as to calculate the arithmetic average.

The results of *AR* and *E[CTDL]* is shown in Exhibit [4.14] and Exhibit [4.15], respectively. And the data was shown in Appendix [2.8] in Appendix 2.

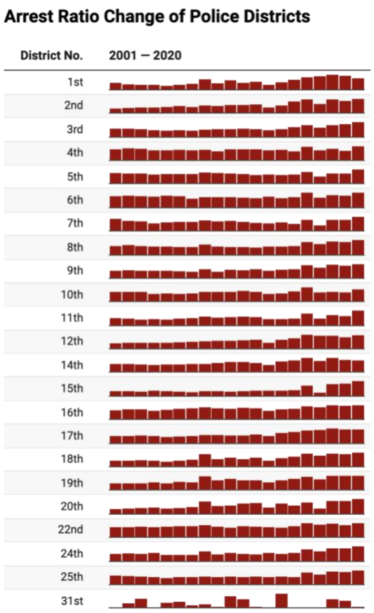

Exhibit [4.14]

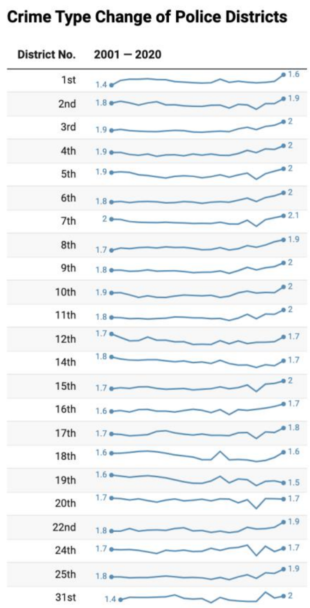

Exhibit [4.15]

(4) Applying Model

After applying the model we established above, we got the *CDL* as shown in Appendix [2.9].

(5) Second Exponential Smoothing Prediction

We first applied the ***Single Exponential Smoothing***, which is the basic of second exponential smoothing, using the formula as:

$$S_{t+1} = \alpha X_t + (1-\alpha) S_t \quad [4.4]$$

We decided the parameter α=0.275 by minimizing the average MSE for all districts.

And we then used the second exponential smoothing method, which is defined as (processing tables are included in Appendix 3):

$$S_t^{(2)} = \alpha S_t^{(1)} + (1-\alpha) S_{t-1}^{(2)} \quad [4.5]$$

$$\begin{cases} a_t = 2S_t^{(1)} - S_t^{(2)} \\ b_t = \frac{a}{1-a}(S_t^{(1)} - S_t^{(2)}) \end{cases} \quad [4.6]$$

$$\hat{Y}_{t+T} = a_t + b_t \cdot T \quad [4.7]$$

Since the coming year is 2022, and second exponential smoothing method is designed for short-term prediction. So we decided to predict the *CDL* of year 2022. And inside the smoothing method, b_t shows the short-term moving trend. So we also got the b_t of year 2020, the last year of the original data.

We put the result on the map of Chicago, using DataWrapper, as shown in Exhibit [4.16].

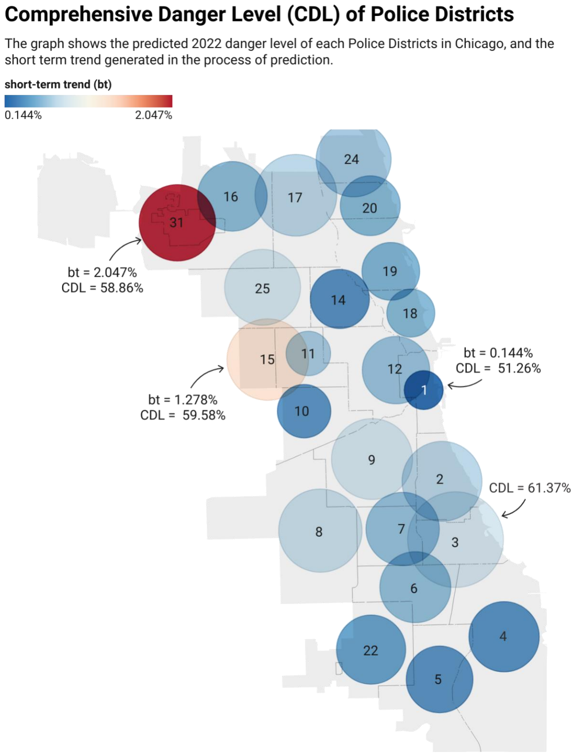

Exhibit [4.16]

The area of the circle shows the 2022 *predicted CDL*, so the larger the circle, the higher the *CDL*. And the color of the circle shows the *short-term trend of CDL*. The redder the color, the fiercer the increasing of *CDL*.

4.5.2. Result

As we can see in Exhibit [4.4], District 3 have the highest *predicted CDL* in 2022. District 31 and District 15 have the largest b_t, and also have relatively high *predicted CDL*. So the three districts are recommended to strengthen police force to reduce *CDL*.

On the other side, circle of District 1 is the smallest, as well as the bluest. So we can infer that District 1 is the safest place to live in Chicago in recent years

5. Conclusion

5.1. Conclusion

In conclusion, we found that our results are consistent with our common knowledge of the distribution of crime. For the results of the time distribution of crime occurrence, we discovered most of the crime happened in the warm season and at the midnight, fewer happened in colder season and early morning. One finding that is different from our common sense is that most districts also reach a peak in the frequency of crime at noon each day. For the results of the crime type distribution, "Theft", "Narcotics" and "Battery" is the most common type, which also in lined with the current state of social security issues. By calculating the danger level in each district and make the prediction, we found the district in

South and West area are more dangerous, and the East middle area is relatively safe. This result is in line with the regional distribution of safety conditions in Chicago that we have collected from Internet. The arrest ratio is highest in the middle area and lowest in the north part of Chicago, which shows the same results with the danger level we calculated, especially in the most dangerous part we found they have the lowest arrest ratio. In response to our findings and predictions, the Chicago Police Department can increase its presence in these future high crime times and areas to prevent them in advance, the Chicago citizens can be more vigilant and take precautions in the high incidence time and location.

5.2. Suggestions

Our data set is very large and comprehensive, with different division of the area in Chicago. However, the "Description" column brought us some trouble, as it contains many commas in the csv file, which make it more difficult to working with these data in the Hadoop tools. Besides, we want to know more information about the police force in Chicago that corresponds to different districts and crime types, so that we can combine information about our results with it together, getting more useful results and suggestions for improving the security in Chicago.

Appendix 1: Codes

Part 1: Analysis of Crime Descriptions

1. **Code for wordcount in descriptions:**

- hdfs dfs -put Chicago_Crimes.csv
- nano crime_mapper.py

```
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    if line.split(",")[6] == '' or line.split(",")[6] == 'Description'\
         or line.split(",")[11] == '' or line.split(",")[17] == '':
       continue
    des = line.split(",")[6]
    district = line.split(",")[11]
    year = line.split(",")[17]

    L_district = ['001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017', \
                  '018','019','020','022','024','025','031']
    if district in L_district:
        words=des.split()
        for word in words:
       print '%s\t%s' %(word.lower(), 1)
```

- nano crime_reducer.py

```
#!/usr/bin/env python

import sys

wordcount={}

for line in sys.stdin:
        word, count = line.strip().split('\t')

        try:
            count=int(count)
        except ValueError:
            continue
        try:
            wordcount[word]=wordcount[word] + count
        except:
            wordcount[word]=count

for word in wordcount.keys():
print '%s\t%s' %(word, wordcount[word])
```

- chmod +x crime_mapper.py
- chmod +x crime_reducer.py

- nano crime_bash.sh

```
#!/bin/bash
 hadoop jar /opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/jars/hadoop-streaming-3.1.1.7.1.7.0-551.jar \
  -Dmapred.reduce.tasks=1 \
  -input /user/xinya.z/Chicago_Crimes.csv \
  -output /user/xinya.z/crime_output \
  -file crime_mapper.py \
  -file crime_reducer.py \
  -mapper "python crime_mapper.py" \
  -reducer "python crime_reducer.py"
```

- bash crime_bash.sh
- hdfs dfs -cat crime_output/part-00000 | sort -k2 -rn | head -20

2. Code for wordclouds:

- nano word_map.py

```
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    if line.split(",")[6] == '' or line.split(",")[6] == 'Description'\
         or line.split(",")[11] == '' or line.split(",")[17] == '':
       continue
```

```
            des = line.split(",")[6]
            district = line.split(",")[11]
            year = line.split(",")[17]
        print '%s\t%s\t%s' % (year, district, des)
```

- nano wordcloud.py

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split("\t")
    year = words[0]
    dis = words[1]
    des = words[2]
    if year == "2020":
       des = des.replace(" ", "~")
  print ('%s' %des)
```

- cat Chicago_Crimes.csv|python word_map.py |python wordcloud.py > sample.txt
- cat sample.txt | shuf -n 10000 >> sample10000.txt
- cat sample.txt | shuf -n 10000 >> sample10000_2.txt
- cat sample.txt | shuf -n 10000 >> sample10000_3.txt
- nano wordcloud.py

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split("\t")
    year = words[0]
    dis = words[1]
    des = words[2]
    if year == "2015":
       des = des.replace(" ", "~")
  print ('%s' %des)
```

- cat Chicago_Crimes.csv|python word_map.py |python wordcloud.py > sample2015.txt
- cat sample2015.txt |shuf -n 10000 >> sample2015_1.txt
- cat sample2015.txt |shuf -n 10000 >> sample2015_2.txt
- cat sample2015.txt |shuf -n 10000 >> sample2015_3.txt
- nano wordcloud.py

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split("\t")
    year = words[0]
    dis = words[1]
    des = words[2]
    if year == "2010":
       des = des.replace(" ", "~")
  print ('%s' %des)
```

- cat Chicago_Crimes.csv|python word_map.py |python wordcloud.py > sample2010.txt
- cat sample2015.txt |shuf -n 10000 >> sample2010_1.txt
- cat sample2015.txt |shuf -n 10000 >> sample2010_2.txt
- cat sample2015.txt |shuf -n 10000 >> sample2010_3.txt
- nano wordcloud.py

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split("\t")
    year = words[0]
    dis = words[1]
    des = words[2]
    if year == "2005":
       des = des.replace(" ", "~")
  print ('%s' %des)
```

- cat Chicago_Crimes.csv|python word_map.py |python wordcloud.py > sample2005.txt
- cat sample2015.txt |shuf -n 10000 >> sample2005_1.txt
- cat sample2015.txt |shuf -n 10000 >> sample2005_2.txt
- cat sample2015.txt |shuf -n 10000 >> sample2005_3.txt

Part 2: Time Distribution of Crime Occurrences

1. Monthly Distribution
    - SELECT substr(`date`,1,2) as month_occurred, count(*) as case_num

- FROM chicago_crimes
- GROUP BY substr(`date`,1,2)
- SORT BY month_occurred DESC

2. Monthly distribution by District
   - SELECT district,substr(`date`,1,2) as month_occurred, count(*) as case_num
   - FROM chicago_crimes
   - GROUP BY substr(`date`,1,2),district
   - SORT BY district ASC, case_num DESC
3. Hourly distribution Total
   - SELECT substr(from_unixtime(unix_timestamp(`date`,'MM/dd/yyyy hh:mm:ss aa'),
   - 'MM-dd-yyyy HH:mm:ss'),12,2) as hour_occured, count(*) as case_num
   - FROM chicago_crimes
   - GROUP BY substr(from_unixtime(unix_timestamp(`date`,'MM/dd/yyyy hh:mm:ss aa'),
   - 'MM-dd-yyyy HH:mm:ss'),12,2)
   - SORT BY case_num
4. Hourly Distribution By District
   - SELECT district,substr(from_unixtime(unix_timestamp(`date`,'MM/dd/yyyy hh:mm:ss aa'),
   - 'MM-dd-yyyy HH:mm:ss'),12,2) as hour_occured, count(*) as case_num
   - FROM chicago_crimes
   - GROUP BY substr(from_unixtime(unix_timestamp(`date`,'MM/dd/yyyy hh:mm:ss aa'),
   - 'MM-dd-yyyy HH:mm:ss'),12,2), district
   - SORT BY district ASC, case_num DESC

Part 3: Crime Type Distribution And Its Average Danger Level

1. Pyspark:

- crime_rdd=sc.textFile("Chicago_Crimes.csv")
- crime_rdd.take(2)

```
>>> crime_rdd.take(2)
[u'ID,Case Number,Date,Block,IUCR,Primary Type,Description,Location Description,Arrest,Domestic,Beat,District,Ward,Community A
rea,FBI Code,X Coordinate,Y Coordinate,Year,Updated On,Latitude,Longitude,Location,degree,,', u'11034701,JA366925,01/01/2001 1
1:00:00 AM,016XX E 86TH PL,1153,DECEPTIVE PRACTICE,FINANCIAL IDENTITY THEFT OVER $ 300,RESIDENCE,FALSE,FALSE,412,4,8,45,11,,,2
001,08/05/2017 03:50:08 PM,,,,1,,']
```

- header=crime_rdd.first()
- crime_rdd1=crime_rdd.filter(lambda line: line != header)
- crime_rdd1.take(2)

```
>>> crime_rdd1.take(2)
[u'11034701,JA366925,01/01/2001 11:00:00 AM,016XX E 86TH PL,1153,DECEPTIVE PRACTICE,FINANCIAL IDENTITY THEFT OVER $ 300,RESIDE
NCE,false,false,0412,004,8,45,11,,,2001,08/05/2017 03:50:08 PM,,,', u'11227287,JB147188,10/08/2017 03:00:00 AM,092XX S RACINE
AVE,0281,CRIM SEXUAL ASSAULT,NON-AGGRAVATED,RESIDENCE,false,false,2222,022,21,73,02,,,2017,02/11/2018 03:57:41 PM,,,']
```

- crime_rdd2=crime_rdd1.map(lambda x: x.split(",")).map(lambda x: [x[11].encode('ascii'), x[5].encode('ascii'), x[0].encode('ascii')])
- crime_rdd2.take(2)

```
>>> crime_rdd2.take(2)
[['004', 'DECEPTIVE PRACTICE', '11034701'], ['022', 'CRIM SEXUAL ASSAULT', '11227287']]
```

- df=sqlContext.createDataFrame(crime_rdd2, ['district', 'crime_type', 'id'])
- df.show(2)

```
+--------+------------------+--------+
|district|        crime_type|      id|
+--------+------------------+--------+
|     004|DECEPTIVE PRACTICE|11034701|
|     022|CRIM SEXUAL ASSAULT|11227287|
+--------+------------------+--------+
only showing top 2 rows
```

- df1=df.groupBy("district", "crime_type").agg({'id': 'count'})
- df1.show(2)

```
>>> df1.show(2)
+--------+------------------+---------+
|district|        crime_type|count(id)|
+--------+------------------+---------+
|     024|      OTHER OFFENSE|    15368|
|     018|CRIMINAL SEXUAL A...|       76|
+--------+------------------+---------+
only showing top 2 rows
```

- df1.sort('count(id)', ascending=False).show(15)

```
>>> df1.sort('count(id)', ascending=False).show(15)
+--------+----------+---------+
|district|crime_type|count(id)|
+--------+----------+---------+
|     018|     THEFT|   133946|
|     011| NARCOTICS|   125767|
|     001|     THEFT|   124051|
|     019|     THEFT|   105922|
|     007|   BATTERY|    99123|
|     012|     THEFT|    97130|
|     008|     THEFT|    92603|
|     011|   BATTERY|    86970|
|     006|   BATTERY|    81482|
|     014|     THEFT|    79499|
|     015| NARCOTICS|    78941|
|     004|   BATTERY|    77922|
|     003|   BATTERY|    77276|
|     008|   BATTERY|    75781|
|     025|     THEFT|    74887|
+--------+----------+---------+
only showing top 15 rows
```

2. MapReduce:

- Hdfs dfs -ls

```
[xiaowen.z@ip-172-31-95-86 ~]$ hdfs dfs -ls
Found 18 items
drwx------   - xiaowen.z xiaowen.z          0 2021-12-07 10:18 .Trash
drwxrwxrwx   - xiaowen.z xiaowen.z          0 2021-11-12 04:17 .scratchdir
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-02 04:01 .sparkStaging
drwx------   - xiaowen.z xiaowen.z          0 2021-12-07 10:17 .staging
-rw-r--r--   3 xiaowen.z xiaowen.z 1684766921 2021-12-07 10:27 Chicago_Crimes.csv
-rw-r--r--   3 xiaowen.z xiaowen.z  245975870 2021-12-07 10:03 Chicago_Crimes_.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-07 10:17 crime1_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-07 10:12 crime_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-12 03:52 flights_output
-rw-r--r--   3 xiaowen.z xiaowen.z         77 2021-11-17 02:36 text.txt
-rw-r--r--   3 xiaowen.z xiaowen.z      34669 2021-11-10 20:48 twitter.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-22 06:43 uc_output
-rw-r--r--   3 xiaowen.z xiaowen.z   69145610 2021-11-22 06:41 used_cars.csv
-rw-r--r--   3 xiaowen.z xiaowen.z     333406 2021-11-22 09:28 used_cars1.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-17 03:17 wc_output
-rw-r--r--   3 xiaowen.z xiaowen.z   19701845 2021-11-17 20:57 youtube.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-17 21:40 youtube_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-18 04:33 yt_wc_output
```

- Nano crime_mapper.py

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
    line = line.strip()
    crime_type = line.split(",")[5]
    district = line.split(",")[11]
    print '%s\t%s' % (district, crime_type)
```

- nano crime_reducer.py

```
#!/usr/bin/env python
import sys
typetal = {}
typecount = {}
D = {'CONCEALED CARRY LICENSE VIOLATION':1.0, 'DECEPTIVE PRACTICE':1.0, 'GAMBLING':1.0, 'INTERFERENCE WITH PUBLIC OFFICER':1.0, \
        'LIQUOR LAW VIOLATION':1.0, 'MOTOR VEHICLE THEFT':1.0, 'NARCOTICS':1.0, 'NON - CRIMINAL':1.0, \
        'NON-CRIMINAL':1.0, 'NON-CRIMINAL (SUBJECT SPECIFIED)':1.0, 'OTHER NARCOTIC VIOLATION':1.0, \
        'OTHER OFFENSE':1.0, 'RITUALISM':1.0, 'PUBLIC PEACE VIOLATION':1.0, 'STALKING':1.0, 'THEFT':1.0, \
        'ARSON':2.0, 'ASSAULT':2.0, 'BURGLARY':2.0, 'CRIMINAL DAMAGE':2.0, 'CRIMINAL TRESPASS':2.0, \
        'INTIMIDATION':2.0, 'PROSTITUTION':2.0, 'ROBBERY':2.0, 'CRIMINAL SEXUAL ASSAULT':3.0, 'BATTERY':3.0, \
        'CRIM SEXUAL ASSAULT':3.0, 'KIDNAPPING':3.0, 'OBSCENITY':3.0, 'OFFENSE INVOLVING CHILDREN':3.0, \
        'PUBLIC INDECENCY':3.0, 'SEX OFFENSE':3.0, 'HOMICIDE':4.0, 'HUMAN TRAFFICKING':4.0, 'WEAPONS VIOLATION':4.0}

L_district=['001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017', \
'018','019','020','022','024','025','031']

for line in sys.stdin:
        line = line.strip().split('\t')
        if len(line) == 2:
                district = line[0]
                crime_type = line[1]
        else:
                continue
        if district not in L_district:
                continue
        if crime_type not in D:
                continue
        try:
                typetal[district] += D[crime_type]
        except:
            typetal[district] = D[crime_type]
      try:
                typecount[district] += 1
        except:
                typecount[district] = 1
for district in typetal:
   print '%s\t%s' % (district, float(typetal[district]/typecount[district]))
```

- cat Chicago_Crimes.csv | python crime_mapper.py | python crime_reducer.py
- nano crime_runmr.sh

```
!/bin/bash
hadoop jar /opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/jars/hadoop-streaming-3.1.1.7.1.7.0-551.jar \
        -Dmapred.reduce.tasks=1 \
        -input /user/xiaowen.z/Chicago_Crimes.csv \
        -output /user/xiaowen.z/crimes_output \
        -file crime_mapper.py \
        -file crime_reducer.py \
        -mapper "python crime_mapper.py" \
   -reducer "python crime_reducer.py"
```

- bash crime_runmr.sh

```
[xiaowen.z@ip-172-31-95-86 ~]$ nano crime_runmr.sh
[xiaowen.z@ip-172-31-95-86 ~]$ bash crime_runmr.sh
crime_runmr.sh: line 1: !/bin/bash: No such file or directory
WARNING: Use "yarn jar" to launch YARN applications.
21/12/07 10:32:03 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [crime_mapper.py, crime_reducer.py] [/opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/jars/hadoop-streaming-3.1.1.7.1.7.0-
551.jar] /tmp/streamjob4582554646583304109.jar tmpDir=null
21/12/07 10:32:04 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-95-86.ec2.internal/172.31.95.86:8032
21/12/07 10:32:04 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-95-86.ec2.internal/172.31.95.86:8032
21/12/07 10:32:04 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /user/xiaowen.z/.staging/job_1638824267190_0083
21/12/07 10:32:04 INFO mapred.FileInputFormat: Total input files to process : 1
21/12/07 10:32:04 INFO mapreduce.JobSubmitter: number of splits:13
21/12/07 10:32:04 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
21/12/07 10:32:04 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1638824267190_0083
21/12/07 10:32:04 INFO mapreduce.JobSubmitter: Executing with tokens: []
21/12/07 10:32:05 INFO conf.Configuration: resource-types.xml not found
21/12/07 10:32:05 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
21/12/07 10:32:05 INFO impl.YarnClientImpl: Submitted application application_1638824267190_0083
21/12/07 10:32:05 INFO mapreduce.Job: The url to track the job: http://ip-172-31-95-86.ec2.internal:8088/proxy/application_1638824267190_0083/
21/12/07 10:32:05 INFO mapreduce.Job: Running job: job_1638824267190_0083
21/12/07 10:32:11 INFO mapreduce.Job: Job job_1638824267190_0083 running in uber mode : false
21/12/07 10:32:11 INFO mapreduce.Job:  map 0% reduce 0%
21/12/07 10:32:21 INFO mapreduce.Job:  map 31% reduce 0%
21/12/07 10:32:29 INFO mapreduce.Job:  map 54% reduce 0%
21/12/07 10:32:30 INFO mapreduce.Job:  map 62% reduce 0%
21/12/07 10:32:38 INFO mapreduce.Job:  map 85% reduce 0%
21/12/07 10:32:39 INFO mapreduce.Job:  map 92% reduce 0%
21/12/07 10:32:45 INFO mapreduce.Job:  map 100% reduce 0%
21/12/07 10:32:55 INFO mapreduce.Job:  map 100% reduce 82%
21/12/07 10:33:01 INFO mapreduce.Job:  map 100% reduce 91%
21/12/07 10:33:05 INFO mapreduce.Job:  map 100% reduce 100%
21/12/07 10:33:06 INFO mapreduce.Job: Job job_1638824267190_0083 completed successfully
21/12/07 10:33:06 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=19996161
                FILE: Number of bytes written=43519338
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1685554991
```

```
[xiaowen.z@ip-172-31-95-86 ~]$ hdfs dfs -ls
Found 19 items
drwx------   - xiaowen.z xiaowen.z          0 2021-12-07 10:18 .Trash
drwxrwxrwx   - xiaowen.z xiaowen.z          0 2021-11-12 04:17 .scratchdir
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-02 04:01 .sparkStaging
drwx------   - xiaowen.z xiaowen.z          0 2021-12-07 10:33 .staging
-rw-r--r--   3 xiaowen.z xiaowen.z 1684766921 2021-12-07 10:27 Chicago_Crimes.csv
-rw-r--r--   3 xiaowen.z xiaowen.z  245975870 2021-12-07 10:03 Chicago_Crimes_.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-07 10:17 crime1_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-07 10:12 crime_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-12-07 10:33 crimes_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-12 03:52 flights_output
-rw-r--r--   3 xiaowen.z xiaowen.z         77 2021-11-17 02:36 text.txt
-rw-r--r--   3 xiaowen.z xiaowen.z      34669 2021-11-10 20:48 twitter.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-22 06:43 uc_output
-rw-r--r--   3 xiaowen.z xiaowen.z   69145610 2021-11-22 06:41 used_cars.csv
-rw-r--r--   3 xiaowen.z xiaowen.z     333406 2021-11-22 09:28 used_cars1.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-17 03:17 wc_output
-rw-r--r--   3 xiaowen.z xiaowen.z   19701845 2021-11-17 20:57 youtube.csv
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-17 21:40 youtube_output
drwxr-xr-x   - xiaowen.z xiaowen.z          0 2021-11-18 04:33 yt_wc_output
```

- `hdfs dfs -cat crimes_output/part-00000 | sort -k2 -rn`

```
[xiaowen.z@ip-172-31-95-86 ~]$ hdfs dfs -cat crimes_output/part-00000 | sort -k2 -rn
007     1.91211309816
005     1.89535593903
003     1.86748918488
004     1.84564957354
006     1.81923382784
010     1.81356886409
009     1.79620766154
002     1.77646805301
008     1.75854521217
022     1.7550670295
025     1.75053774261
015     1.74918279819
011     1.72551960046
024     1.72259302759
017     1.69560241784
014     1.6758961636
020     1.67397409565
012     1.66202614007
016     1.65222210005
031     1.58706467662
019     1.58324283284
018     1.50685391426
001     1.44311679508
```

Part 4: Arrest Ratio Analysis

1. Upload csv file to server

`scp -i Desktop/S_keypair.pem Desktop/Chicago_Crimes.csv jingyi.zhu@18.206.158.228:Chicago_Crimes.csv`

2. Copy the file to HDFS

`hdfs dfs -put Chicago_Crimes.csv`

3. NANO the Python File

   a) Write Map Function

      `nano arrest_mapper.py`

   b) Mapper

```
#!/usr/bin/env python

import sys

L_arrest = ['Arrest','true','false']

L_district = ['District','001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017',\
'018','019','020','022','024','025','031']

for line in sys.stdin:

        line = line.strip().split(",")

        i = 7
        j = 10

        arrest = line[i]
        district = line[j]

        while arrest not in L_arrest:
                i += 1
                if i >= len(line):
                        break
                arrest = line[i]

        while district not in L_district:
                j += 1
                if j >= len(line):
                        break
                district = line[j]

        print '%s\t%s' % (district, arrest)
```

c) **Write Reduce Function**

```
nano arrest_reducer.py
```

d) **Reducer:**

```
#!/usr/bin/env python

import sys

arrtal = {}
arrcount = {}

D_arrest = {'true':1.0,'false':0.0}

L_district = ['001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017', \
'018','019','020','022','024','025','031']

for line in sys.stdin:

        line = line.strip().split('\t')

        if len(line) == 2:
                district = line[0]
                arrest = line[1]
        else:
                continue

        if district not in L_district:
                continue
        if arrest not in D_arrest:
                continue

        try:
                arrtal[district] += D_arrest[arrest]
        except:
                arrtal[district] = D_arrest[arrest]
        try:
                arrcount[district] += 1
        except:
                arrcount[district] = 1


for district in arrtal:
        print '%s,%s' % (district, float(arrtal[district]/arrcount[district]))
```

4. **Make both 'py' executable**

```
chmod +x arrest_mapper.py
chmod +x arrest_reducer.py
```

5. **Execute both codes using standard IO**

   a) **Create file**

   ```
   nano arrest.sh
   ```

   b) **Code**

   ```
   #!/bin/bash
   hadoop jar /opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/jars/hadoop-streaming-3.1.1.7.1.7.0-551.jar \
           -Dmapred.reduce.tasks=1 \
           -input /user/jingyi.zhu/Chicago_Crimes.csv \
           -output /user/jingyi.zhu/arrest_result \
           -file arrest_mapper.py \
           -file arrest_reducer.py \
           -mapper 'python arrest_mapper.py' \
           -reducer 'python arrest_reducer.py'
   ```

6. **Run bash**

   ```
   bash arrest.sh
   ```

7. **Copy to local**

   ```
   hdfs dfs -cat arrest_result/part-00000 > ~/arrest_ratio.csv
   ```

8. **Show the result**

   ```
   cat arrest_ratio.csv
   ```

Par 5: Prediction of Comprehensive Danger Level

1. Step 1 Code:

Upload csv file to server

```
scp -i Desktop/S_keypair.pem Desktop/Chicago_Crimes.csv jingyi.zhu@18.206.158.228:Chicago_Crimes.csv
```

Copy the file to HDFS

```
hdfs dfs -put Chicago_Crimes.csv
```

NANO the Python File

   a) **Write Map Function**

   ```
   nano yr_arrest_mapper.py
   ```

   b) **Mapper**

   ```
   #!/usr/bin/env python
   import sys
   L_arrest = ['Arrest','true','false']

   L_district = ['District','001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017',\
   '018','019','020','022','024','025','031']

   L_year = ['Date','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014',\
   '2015','2016','2017','2018','2019','2020']

   for line in sys.stdin:

           line = line.strip().split(",")

           i = 7
           j = 10
           k = 2

           arrest = line[i]
           district = line[j]
           year = line[k].split('/')[0]

           while arrest not in L_arrest:
                   i += 1
                   if i >= len(line):
                           break
                   arrest = line[i]

           while district not in L_district:
                   j += 1
                   if j >= len(line):
                           break
                   district = line[j]

           while year not in L_year:
                   k += 1
                   if k >= len(line):
                           break
                   year = line[k].split('/')[0]
   ```

```
        print '%s\t%s\t%s' % (year, district, arrest)
```

c) **Write Reduce Function**
```
nano yr_arrest_reducer.py
```

d) **Reducer:**
```
#!/usr/bin/env python

import sys

arrtal = {}
arrcount = {}

D_arrest = {'true':1.0,'false':0.0}
L_district = ['001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017', \
'018','019','020','022','024','025','031']

L_year = ['2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014',\
'2015','2016','2017','2018','2019','2020']

for line in sys.stdin:

        line = line.strip().split('\t')

        if len(line) == 3:
                year = line[0]
                district = line[1]
                arrest = line[2]
        else:
                continue

        if year not in L_year:
                continue
        if district not in L_district:
                continue
        if arrest not in D_arrest:
                continue

        year = int(year)

        try:
                arrtal[(year,district)] += D_arrest[arrest]
        except:
                arrtal[(year,district)] = D_arrest[arrest]
        try:
                arrcount[(year,district)] += 1
        except:
                arrcount[(year,district)] = 1

for year_district in arrtal:
        print '%s,%s,%s' % (year_district[0], year_district[1], float(arrtal[year_district]/arrcount[year_district]))
```

e) **Make both 'py' executable**
```
chmod +x yr_arrest_mapper.py

chmod +x yr_arrest_reducer.py
```

f) **Execute both codes using standard IO**

g) **Create file**
```
Nano yr_arrest.sh
```

h) **Code**
```
#!/bin/bash
hadoop jar /opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/jars/hadoop-streaming-3.1.1.7.1.7.0-551.jar \
        -Dmapred.reduce.tasks=1 \
        -input /user/jingyi.zhu/Chicago_Crimes.csv \
        -output /user/jingyi.zhu/arrest_ratio \
        -file yr_arrest_mapper.py \
        -file yr_arrest_reducer.py \
        -mapper 'python yr_arrest_mapper.py' \
        -reducer 'python yr_arrest_reducer.py'
```

i) **Run bash**
```
bash yr_arrest.sh
```

j) **Copy to local**
```
hdfs dfs -cat arrest_ratio/part-00000 > ~/yr_arrest_ratio.csv
```

k) **Copy to Desktop**
```
scp -i Desktop/S_keypair.pem jingyi.zhu@18.206.158.228:yr_arrest_ratio.csv ~/Desktop/
```

2. Step 2 Code:

NANO the Python File

a) Write Map Function
   nano yr_crime_mapper.py

b) Mapper

```
#!/usr/bin/env python

import sys

L_crime_type = ['Primary Type', 'CONCEALED CARRY LICENSE VIOLATION', 'DECEPTIVE PRACTICE', 'GAMBLING', \
        'INTERFERENCE WITH PUBLIC OFFICER', 'LIQUOR LAW VIOLATION', 'MOTOR VEHICLE THEFT', 'NARCOTICS', \
        'NON - CRIMINAL', 'NON-CRIMINAL', 'NON-CRIMINAL (SUBJECT SPECIFIED)', 'OTHER NARCOTIC VIOLATION', \
        'OTHER OFFENSE', 'RITUALISM', 'PUBLIC PEACE VIOLATION', 'STALKING', 'THEFT', \
        'ARSON', 'ASSAULT', 'BURGLARY', 'CRIMINAL DAMAGE', 'CRIMINAL TRESPASS', \
        'INTIMIDATION', 'PROSTITUTION', 'ROBBERY', 'CRIMINAL SEXUAL ASSAULT', 'BATTERY', \
        'CRIM SEXUAL ASSAULT', 'KIDNAPPING', 'OBSCENITY', 'OFFENSE INVOLVING CHILDREN', \
        'PUBLIC INDECENCY', 'SEX OFFENSE', 'HOMICIDE', 'HUMAN TRAFFICKING', 'WEAPONS VIOLATION']

L_district = ['District','001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017',\
'018','019','020','022','024','025','031']

L_year = ['Date','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014',\
'2015','2016','2017','2018','2019','2020']

for line in sys.stdin:

        line = line.strip().split(",")

        i = 5
        j = 10
        k = 2

        crime_type = line[i]
        district = line[j]
        year = line[k].split('/')[0]

        while crime_type not in L_crime_type:
                i += 1
                if i >= len(line):
                        break
                crime_type = line[i]

        while district not in L_district:
                j += 1
                if j >= len(line):
                        break
                district = line[j]

        while year not in L_year:
                k += 1
                if k >= len(line):
                        break
                year = line[k].split('/')[0]

        print '%s\t%s\t%s' % (year, district, crime_type)
```

c) Write Reduce Function
   nano yr_crime_reducer.py

d) Reducer:

```
#!/usr/bin/env python

import sys

typetal = {}
typecount = {}

D = {'CONCEALED CARRY LICENSE VIOLATION':1.0, 'DECEPTIVE PRACTICE':1.0, 'GAMBLING':1.0, \
        'INTERFERENCE WITH PUBLIC OFFICER':1.0, 'LIQUOR LAW VIOLATION':1.0, 'MOTOR VEHICLE THEFT':1.0, \
   'NARCOTICS':1.0, 'NON - CRIMINAL':1.0, 'NON-CRIMINAL':1.0, 'NON-CRIMINAL (SUBJECT SPECIFIED)':1.0, \
        'OTHER NARCOTIC VIOLATION':1.0, 'OTHER OFFENSE':1.0, 'RITUALISM':1.0, 'PUBLIC PEACE VIOLATION':1.0, \
        'STALKING':1.0, 'THEFT':1.0, 'ARSON':2.0, 'ASSAULT':2.0, 'BURGLARY':2.0, 'CRIMINAL DAMAGE':2.0, \
        'CRIMINAL TRESPASS':2.0, 'INTIMIDATION':2.0, 'PROSTITUTION':2.0, 'ROBBERY':2.0, 'CRIMINAL SEXUAL ASSAULT':3.0, \
        'BATTERY':3.0, 'CRIM SEXUAL ASSAULT':3.0, 'KIDNAPPING':3.0, 'OBSCENITY':3.0, 'OFFENSE INVOLVING CHILDREN':3.0, \
        'PUBLIC INDECENCY':3.0, 'SEX OFFENSE':3.0, 'HOMICIDE':4.0, 'HUMAN TRAFFICKING':4.0, 'WEAPONS VIOLATION':4.0}

L_district = ['001','002','003','004','005','006','007','008','009','010','011','012','014','015','016','017', \
```

```
'018','019','020','022','024','025','031']

L_year = ['2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014',\
'2015','2016','2017','2018','2019','2020']

for line in sys.stdin:
        line = line.strip().split('\t')
        if len(line) == 3:
                year = line[0]
                district = line[1]
                crime_type = line[2]
        else:
                continue
        if year not in L_year:
                continue
        if district not in L_district:
                continue
        if crime_type not in D:
                continue
        year = int(year)
        try:
                typetal[(year,district)] += D[crime_type]
        except:
                typetal[(year,district)] = D[crime_type]
        try:
                typecount[(year,district)] += 1
        except:
                typecount[(year,district)] = 1
for year_district in typetal:
        print '%s,%s,%s' % (year_district[0], year_district[1], float(typetal[year_district]/typecount[year_district]))
```

e) **Make both 'py' executable**

```
chmod +x yr_crime_mapper.py
chmod +x yr_crime_reducer.py
```

f) **Execute both codes using standard IO**

```
#!/bin/bash
hadoop jar /opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/jars/hadoop-streaming-3.1.1.7.1.7.0-551.jar \
        -Dmapred.reduce.tasks=1 \
        -input /user/jingyi.zhu/Chicago_Crimes.csv \
        -output /user/jingyi.zhu/crime_type \
        -file yr_crime_mapper.py \
        -file yr_crime_reducer.py \
        -mapper 'python yr_crime_mapper.py' \
        -reducer 'python yr_crime_reducer.py'
```

g) **Run bash**

```
bash yr_crime.sh
```

h) **Copy to local**

```
hdfs dfs -cat crime_type/part-00000 > ~/yr_crime_type.csv
```

i) **Copy to Desktop**

```
scp -i Desktop/S_keypair.pem jingyi.zhu@18.206.158.228:yr_crime_type.csv ~/Desktop/
```

Appendix 2

2.1 Top 20 Words in Crime Descriptions

```
simple  1328130
to      970439
$500    934061
domestic        592821
under   580342
and     580157
battery 554561
poss:   548544
over    454524
vehicle 439036
property        397754
entry   394876
or      316715
automobile      293470
cannabis        293085
theft   290528
forcible        287853
30gms   287793
less    280641
telephone       246668
```

2.2 Requirement of wordclouds.com

Join words into phrases using the ~ character: **cup~of~tea**

2.3 Sample Results after Changing the Format

```
OVER~$500
OVER~$500
AGG~CRIM~SEX~ABUSE~FAM~MEMBER
AGG~SEX~ASSLT~OF~CHILD~FAM~MBR
FORGERY
AGG~CRIM~SEX~ABUSE~FAM~MEMBER
NON-AGGRAVATED
FINANCIAL~IDENTITY~THEFT~OVER~$~300
SIMPLE
AUTOMOBILE
SEX~ASSLT~OF~CHILD~BY~FAM~MBR
$500~AND~UNDER
FORGERY
SEX~OFFENDER:~FAIL~TO~REGISTER
DOMESTIC~BATTERY~SIMPLE
EMBEZZLEMENT
AGGRAVATED~SEXUAL~ASSAULT~OF~CHILD~BY~FAMILY~MEMBER
FINANCIAL~IDENTITY~THEFT~OVER~$~300
FINANCIAL~IDENTITY~THEFT~OVER~$~300
PREDATORY
PREDATORY
FINANCIAL~IDENTITY~THEFT~OVER~$~300
AGG~CRIM~SEX~ABUSE~FAM~MEMBER
FROM~BUILDING
DOMESTIC~BATTERY~SIMPLE
DOMESTIC~BATTERY~SIMPLE
TO~LAND
$500~AND~UNDER
TO~LAND
```

2.4

	month_occurred	case_num
1	07	655671
2	08	649052
3	05	643964
4	06	636854
5	10	614766
6	09	608207
7	03	594520
8	04	592885
9	01	568203
10	11	553619
11	12	525558
12	02	500400

2.5

district	month_occurred	case_num
1	8	26295
1	7	25958
1	6	24956
1	5	24406
1	10	24283
1	9	23745
1	1	23411
1	3	23301
1	11	22590
1	4	22408
1	12	22107
1	2	20597

2.6

	hour_occured	case_num
1	05	96230
2	06	113258
3	04	115637
4	03	153276
5	07	162516
6	02	189993
7	01	226137
8	08	241916
9	10	301386
10	09	307849
11	11	316634
12	23	321118
13	13	340137
14	16	360184
15	14	362088
16	17	366095
17	15	380639
18	22	388318
19	18	392854
20	21	393811
21	00	394345
22	20	404642
23	19	406143
24	12	408493

district	hour_occu	case_num
1	12	22477
1	13	19488
1	14	18886
1	17	18796
1	15	18765
1	16	18481
1	18	16876
1	11	15318
1	19	15226
1	9	14687
1	10	14067
1	20	12760
1	21	10855
1	8	10809
1	0	9720
1	22	9483

2.7 Danger Level Assignment

Primary Type	Danger Degree		
CONCEALED CARRY LICENSE VIOLATION	1	1	Non-criminal, not involving personal safety
DECEPTIVE PRACTICE	1	2	Involving minor personal injury
GAMBLING	1	3	Involving moderate personal injury
INTERFERENCE WITH PUBLIC OFFICER	1	4	Involved in serious personal injury resulting in death
LIQUOR LAW VIOLATION	1		
MOTOR VEHICLE THEFT	1		
NARCOTICS	1		
NON - CRIMINAL	1		
NON-CRIMINAL	1		
NON-CRIMINAL (SUBJECT SPECIFIED)	1		
OTHER NARCOTIC VIOLATION	1		
OTHER OFFENSE	1		
PUBLIC PEACE VIOLATION	1		
RITUALISM	1		
STALKING	1		
THEFT	1		
ARSON	2		
ASSAULT	2		
BURGLARY	2		
CRIMINAL DAMAGE	2		
CRIMINAL SEXUAL ASSAULT	2		
CRIMINAL TRESPASS	2		
INTIMIDATION	2		

PROSTITUTION	2
ROBBERY	2
BATTERY	3
CRIM SEXUAL ASSAULT	3
KIDNAPPING	3
OBSCENITY	3
OFFENSE INVOLVING CHILDREN	3
PUBLIC INDECENCY	3
SEX OFFENSE	3
HOMICIDE	4
HUMAN TRAFFICKING	4
WEAPONS VIOLATION	4

2.8 MapReduce Result

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
2001	30%	37%	26%	22%	24%	26%	23%	24%	31%	32%	42%	31%	23%	41%	22%	24%	31%	29%	34%	23%	25%	27%	
2002	35%	33%	26%	21%	25%	25%	27%	22%	30%	31%	44%	30%	24%	42%	21%	25%	30%	29%	34%	23%	24%	27%	22%
2003	37%	33%	27%	22%	25%	26%	27%	24%	31%	32%	46%	30%	24%	43%	21%	24%	29%	28%	32%	22%	25%	29%	9%
2004	37%	33%	30%	25%	27%	27%	33%	25%	31%	37%	45%	30%	25%	40%	23%	25%	31%	25%	29%	23%	23%	30%	80%
2005	41%	30%	31%	24%	26%	25%	30%	25%	30%	36%	46%	29%	25%	41%	22%	27%	34%	28%	33%	22%	27%	33%	20%
2006	38%	28%	30%	24%	27%	26%	29%	26%	32%	36%	44%	27%	24%	44%	19%	27%	31%	25%	31%	22%	28%	30%	17%
2007	34%	31%	31%	24%	27%	30%	28%	26%	34%	35%	42%	27%	24%	45%	19%	25%	28%	26%	28%	21%	28%	30%	33%
2008	22%	26%	28%	23%	23%	28%	26%	21%	27%	32%	39%	25%	23%	42%	17%	22%	16%	14%	17%	19%	19%	29%	27%
2009	33%	27%	28%	26%	25%	28%	27%	25%	33%	32%	41%	24%	21%	42%	18%	22%	25%	21%	25%	23%	26%	30%	58%
2010	24%	24%	27%	22%	26%	28%	27%	26%	29%	32%	40%	24%	19%	43%	20%	24%	22%	18%	24%	24%	23%	29%	4%
2011	30%	25%	26%	23%	26%	29%	29%	27%	29%	30%	39%	23%	19%	41%	17%	23%	25%	21%	21%	22%	25%	29%	10%
2012	27%	23%	27%	22%	28%	30%	31%	27%	28%	32%	39%	22%	21%	38%	19%	21%	23%	21%	20%	22%	26%	29%	57%
2013	37%	33%	30%	25%	27%	27%	33%	25%	31%	37%	45%	30%	25%	40%	23%	25%	31%	25%	29%	23%	23%	30%	80%
2014	28%	26%	27%	23%	29%	29%	30%	25%	29%	34%	45%	22%	18%	40%	21%	18%	23%	21%	20%	24%	25%	28%	0%
2015	23%	18%	23%	27%	27%	28%	34%	24%	27%	30%	43%	18%	16%	38%	20%	16%	20%	19%	25%	22%	23%	29%	73%
2016	19%	15%	19%	18%	23%	21%	25%	16%	20%	24%	33%	13%	12%	26%	15%	14%	14%	15%	18%	17%	15%	20%	60%
2017	17%	23%	24%	25%	32%	29%	39%	20%	24%	32%	44%	14%	17%	47%	17%	13%	18%	19%	33%	19%	20%	24%	85%
2018	15%	15%	19%	22%	25%	24%	25%	16%	19%	31%	35%	14%	12%	24%	14%	11%	14%	13%	16%	17%	14%	19%	10%
2019	16%	17%	18%	25%	25%	25%	25%	17%	21%	32%	38%	16%	15%	22%	14%	12%	16%	13%	16%	20%	14%	19%	14%
2020	20%	14%	15%	18%	19%	21%	20%	14%	18%	27%	28%	14%	16%	19%	14%	12%	13%	13%	13%	17%	11%	17%	50%

Arrest Ratio of each district for each year from 2001–2020

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
2001	1.35	1.80	1.87	1.87	1.95	1.80	2.00	1.73	1.81	1.86	1.76	1.75	1.76	1.75	1.64	1.70	1.55	1.63	1.71	1.75	1.73	1.76	
2002	1.47	1.83	1.89	1.87	1.96	1.82	1.99	1.76	1.81	1.87	1.76	1.71	1.72	1.79	1.65	1.69	1.55	1.62	1.71	1.75	1.73	1.76	1.44
2003	1.50	1.81	1.87	1.84	1.95	1.79	1.94	1.75	1.79	1.82	1.73	1.67	1.71	1.76	1.64	1.68	1.56	1.61	1.69	1.78	1.73	1.75	1.82
2004	1.50	1.78	1.86	1.83	1.91	1.81	1.92	1.77	1.80	1.75	1.73	1.67	1.70	1.82	1.67	1.69	1.58	1.62	1.70	1.75	1.72	1.75	1.80
2005	1.51	1.81	1.85	1.85	1.89	1.82	1.91	1.77	1.82	1.79	1.71	1.71	1.71	1.83	1.67	1.69	1.59	1.62	1.68	1.76	1.71	1.75	1.80
2006	1.49	1.82	1.85	1.81	1.87	1.81	1.91	1.76	1.79	1.76	1.73	1.68	1.71	1.76	1.65	1.73	1.58	1.61	1.65	1.77	1.70	1.76	1.83
2007	1.48	1.78	1.86	1.84	1.85	1.79	1.92	1.78	1.80	1.75	1.74	1.68	1.69	1.73	1.65	1.74	1.55	1.60	1.68	1.76	1.73	1.77	1.89
2008	1.44	1.78	1.87	1.84	1.88	1.79	1.91	1.77	1.80	1.79	1.78	1.66	1.68	1.74	1.64	1.71	1.52	1.58	1.70	1.78	1.72	1.76	2.18
2009	1.43	1.77	1.86	1.82	1.89	1.82	1.90	1.77	1.78	1.81	1.77	1.66	1.69	1.76	1.65	1.70	1.51	1.56	1.67	1.79	1.73	1.75	1.67
2010	1.41	1.76	1.84	1.85	1.88	1.80	1.89	1.76	1.76	1.80	1.75	1.63	1.66	1.71	1.67	1.69	1.49	1.54	1.68	1.75	1.71	1.75	1.56
2011	1.40	1.76	1.84	1.83	1.86	1.78	1.89	1.74	1.77	1.79	1.75	1.63	1.67	1.73	1.66	1.70	1.45	1.54	1.70	1.76	1.72	1.74	1.62
2012	1.41	1.78	1.85	1.85	1.88	1.80	1.88	1.74	1.77	1.81	1.72	1.63	1.65	1.75	1.63	1.68	1.45	1.57	1.67	1.74	1.74	1.72	1.00
2013	1.50	1.78	1.86	1.83	1.91	1.81	1.92	1.77	1.80	1.75	1.73	1.67	1.70	1.82	1.67	1.69	1.58	1.62	1.70	1.75	1.72	1.75	1.80
2014	1.42	1.74	1.85	1.83	1.87	1.80	1.87	1.73	1.75	1.78	1.67	1.64	1.65	1.75	1.60	1.68	1.45	1.55	1.70	1.73	1.75	1.71	1.40
2015	1.45	1.79	1.90	1.86	1.91	1.84	1.87	1.75	1.80	1.87	1.74	1.67	1.62	1.79	1.67	1.71	1.45	1.59	1.64	1.75	1.75	1.76	1.27
2016	1.42	1.78	1.92	1.91	1.94	1.89	1.97	1.79	1.87	1.91	1.86	1.64	1.62	1.89	1.66	1.72	1.44	1.58	1.69	1.80	1.77	1.80	1.20
2017	1.41	1.72	1.88	1.87	1.83	1.82	1.81	1.77	1.82	1.86	1.72	1.65	1.58	1.64	1.67	1.65	1.42	1.52	1.55	1.78	1.68	1.74	1.20
2018	1.42	1.80	1.94	1.93	1.93	1.90	1.99	1.80	1.86	1.88	1.86	1.65	1.63	1.93	1.68	1.73	1.44	1.56	1.71	1.78	1.76	1.83	2.60
2019	1.44	1.80	1.96	1.92	1.97	1.94	2.04	1.84	1.88	1.87	1.85	1.66	1.61	1.95	1.71	1.72	1.48	1.57	1.70	1.80	1.71	1.82	1.71
2020	1.61	1.86	2.02	1.98	2.01	2.00	2.09	1.87	1.95	2.01	1.98	1.71	1.69	2.05	1.74	1.77	1.57	1.55	1.70	1.86	1.74	1.89	2.00

Average Crime Type Danger Level of each district for each year from 2001–2020

2.9

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
2001	46%	51%	56%	57%	58%	55%	59%	54%	53%	54%	49%	52%	55%	49%	53%	54%	49%	51%	50%	55%	54%	54%	
2002	46%	53%	56%	58%	58%	55%	58%	55%	53%	54%	48%	52%	54%	49%	54%	53%	49%	51%	51%	55%	54%	54%	50%
2003	46%	53%	55%	57%	58%	55%	57%	55%	53%	53%	47%	51%	54%	48%	54%	53%	50%	51%	51%	56%	54%	53%	61%
2004	46%	52%	54%	56%	56%	55%	54%	54%	53%	50%	47%	51%	53%	50%	53%	53%	49%	52%	52%	55%	54%	52%	37%
2005	45%	54%	54%	56%	56%	55%	55%	54%	54%	51%	46%	52%	53%	50%	54%	52%	48%	51%	50%	55%	53%	52%	57%
2006	46%	54%	54%	56%	56%	55%	55%	54%	53%	51%	48%	52%	54%	48%	54%	53%	49%	52%	51%	56%	52%	53%	58%
2007	47%	53%	54%	56%	55%	53%	56%	54%	52%	51%	48%	52%	54%	47%	54%	54%	50%	51%	52%	56%	53%	53%	54%
2008	50%	54%	55%	56%	57%	54%	57%	56%	54%	52%	50%	53%	54%	49%	55%	55%	53%	55%	56%	57%	56%	53%	61%
2009	46%	54%	55%	55%	57%	54%	56%	54%	52%	53%	49%	53%	54%	49%	55%	54%	50%	52%	53%	56%	53%	53%	42%
2010	49%	55%	55%	57%	56%	54%	56%	54%	53%	53%	49%	53%	55%	47%	55%	54%	51%	53%	53%	55%	54%	53%	58%
2011	47%	54%	55%	56%	55%	53%	55%	53%	53%	53%	49%	53%	55%	48%	55%	54%	49%	52%	55%	55%	54%	53%	57%
2012	48%	55%	55%	57%	55%	53%	54%	53%	54%	53%	49%	53%	54%	50%	54%	54%	50%	52%	55%	55%	54%	52%	31%
2013	46%	52%	54%	56%	56%	55%	54%	54%	53%	50%	47%	51%	53%	50%	53%	53%	49%	52%	52%	55%	54%	52%	37%
2014	48%	54%	55%	56%	55%	54%	55%	54%	53%	52%	46%	53%	55%	49%	53%	55%	50%	52%	55%	54%	54%	52%	57%
2015	50%	57%	57%	55%	56%	55%	53%	55%	54%	54%	48%	55%	55%	51%	54%	57%	51%	54%	52%	55%	55%	53%	30%
2016	51%	58%	59%	59%	58%	58%	58%	58%	58%	57%	53%	56%	56%	56%	56%	57%	53%	55%	55%	58%	58%	57%	33%
2017	51%	55%	57%	56%	53%	54%	50%	56%	55%	54%	47%	56%	54%	45%	56%	56%	51%	52%	48%	57%	55%	54%	25%
2018	52%	58%	59%	58%	57%	57%	58%	58%	58%	54%	53%	56%	56%	58%	57%	58%	52%	55%	56%	57%	58%	58%	73%
2019	52%	58%	60%	57%	58%	57%	59%	59%	58%	54%	52%	56%	55%	58%	57%	58%	53%	55%	56%	57%	57%	57%	57%
2020	53%	60%	62%	60%	61%	60%	62%	60%	60%	58%	57%	57%	56%	61%	58%	59%	55%	55%	57%	59%	59%	59%	50%

Appendix 3: Second Exponential Smoothing Process

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
2001	46%	51%	56%	57%	58%	55%	59%	54%	53%	54%	49%	52%	55%	49%	53%	54%	49%	51%	50%	55%	54%	54%	
2002	46%	51%	56%	57%	58%	55%	59%	54%	53%	54%	49%	52%	55%	49%	53%	54%	49%	51%	50%	55%	54%	54%	50%
2003	46%	51%	56%	57%	58%	55%	59%	54%	53%	54%	48%	52%	55%	49%	54%	54%	49%	51%	50%	55%	54%	54%	50%
2004	46%	52%	56%	57%	58%	55%	58%	54%	53%	54%	48%	52%	54%	49%	54%	53%	49%	51%	51%	55%	54%	53%	53%
2005	46%	52%	55%	57%	57%	55%	57%	54%	53%	53%	48%	52%	54%	49%	53%	53%	49%	51%	51%	55%	54%	53%	48%
2006	46%	52%	55%	56%	57%	55%	56%	54%	53%	52%	47%	52%	54%	49%	54%	53%	49%	51%	51%	55%	54%	53%	51%
2007	46%	53%	55%	56%	57%	55%	56%	54%	53%	52%	47%	52%	54%	49%	54%	53%	49%	51%	51%	55%	53%	53%	53%
2008	46%	53%	55%	56%	56%	54%	56%	54%	53%	52%	48%	52%	54%	49%	54%	53%	49%	51%	51%	55%	53%	53%	53%
2009	47%	53%	55%	56%	56%	54%	56%	55%	53%	52%	48%	52%	54%	49%	54%	54%	50%	52%	52%	56%	54%	53%	55%
2010	47%	53%	55%	56%	56%	54%	56%	55%	53%	52%	49%	52%	54%	49%	54%	54%	50%	52%	53%	56%	54%	53%	51%
2011	47%	54%	55%	56%	56%	54%	56%	54%	53%	52%	49%	52%	54%	48%	54%	54%	51%	53%	53%	55%	54%	53%	53%
2012	47%	54%	55%	56%	56%	54%	56%	54%	53%	53%	49%	53%	54%	48%	55%	54%	50%	52%	53%	55%	54%	53%	54%
2013	47%	54%	55%	56%	56%	54%	55%	54%	53%	53%	49%	53%	54%	49%	55%	54%	50%	52%	54%	55%	54%	53%	48%
2014	47%	54%	55%	56%	56%	54%	55%	54%	53%	52%	48%	52%	54%	49%	54%	54%	50%	52%	53%	55%	54%	53%	45%
2015	47%	54%	55%	56%	56%	54%	55%	54%	53%	52%	48%	53%	54%	49%	54%	54%	50%	52%	54%	55%	54%	53%	48%
2016	48%	55%	56%	56%	56%	54%	54%	54%	53%	53%	48%	53%	54%	50%	54%	55%	50%	53%	53%	55%	54%	53%	43%
2017	49%	56%	57%	57%	56%	55%	55%	55%	55%	54%	49%	54%	55%	51%	55%	56%	51%	53%	54%	56%	55%	54%	40%
2018	49%	55%	57%	57%	55%	55%	54%	55%	55%	54%	49%	55%	55%	50%	55%	56%	51%	53%	52%	56%	55%	54%	36%
2019	50%	56%	57%	57%	56%	55%	55%	56%	56%	54%	50%	55%	55%	52%	55%	57%	51%	54%	53%	56%	56%	55%	46%
2020	51%	57%	58%	57%	56%	56%	56%	57%	56%	54%	50%	55%	55%	54%	56%	57%	52%	54%	54%	56%	56%	56%	49%

Appendix [3.1] Single Exponential Smoothing Result

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
2001	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
2002	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	0.02%	0.02%	0.00%	0.00%	0.01%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2003	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.04%	0.00%	0.00%	0.01%	0.03%	0.01%	0.01%	0.01%	0.00%	0.00%	0.01%	0.00%	0.00%	0.01%	0.00%	0.01%	1.13%
2004	0.00%	0.00%	0.03%	0.03%	0.03%	0.00%	0.13%	0.00%	0.00%	0.11%	0.01%	0.00%	0.01%	0.03%	0.00%	0.00%	0.00%	0.01%	0.02%	0.00%	0.00%	0.01%	2.64%
2005	0.01%	0.03%	0.03%	0.00%	0.01%	0.00%	0.03%	0.00%	0.00%	0.02%	0.02%	0.00%	0.00%	0.01%	0.00%	0.01%	0.01%	0.00%	0.00%	0.00%	0.02%	0.03%	0.68%
2006	0.00%	0.05%	0.01%	0.01%	0.01%	0.00%	0.01%	0.00%	0.00%	0.04%	0.00%	0.00%	0.00%	0.02%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.02%	0.00%	0.58%
2007	0.01%	0.00%	0.00%	0.00%	0.02%	0.04%	0.00%	0.00%	0.01%	0.01%	0.01%	0.00%	0.00%	0.04%	0.00%	0.00%	0.01%	0.00%	0.02%	0.00%	0.00%	0.00%	0.01%
2008	0.15%	0.02%	0.00%	0.00%	0.01%	0.00%	0.00%	0.02%	0.02%	0.01%	0.06%	0.00%	0.00%	0.00%	0.01%	0.01%	0.17%	0.14%	0.25%	0.02%	0.06%	0.00%	0.57%
2009	0.01%	0.00%	0.00%	0.02%	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.81%
2010	0.04%	0.01%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.01%	0.02%	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.00%	0.00%	0.43%
2011	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.01%	0.01%	0.00%	0.01%	0.01%	0.00%	0.01%	0.00%	0.01%	0.00%	0.02%	0.00%	0.04%	0.00%	0.00%	0.00%	0.15%
2012	0.00%	0.01%	0.00%	0.00%	0.01%	0.00%	0.02%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.02%	0.00%	0.00%	0.00%	0.00%	0.02%	0.00%	0.00%	0.00%	5.46%
2013	0.02%	0.04%	0.01%	0.01%	0.00%	0.01%	0.01%	0.00%	0.00%	0.06%	0.04%	0.02%	0.01%	0.03%	0.02%	0.01%	0.00%	0.00%	0.03%	0.00%	0.00%	0.00%	1.26%
2014	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.05%	0.01%	0.01%	0.00%	0.01%	0.02%	0.00%	0.00%	0.03%	0.01%	0.00%	0.00%	1.41%
2015	0.06%	0.13%	0.06%	0.01%	0.00%	0.01%	0.04%	0.00%	0.02%	0.06%	0.00%	0.06%	0.01%	0.02%	0.00%	0.06%	0.01%	0.02%	0.01%	0.00%	0.01%	0.00%	3.16%
2016	0.09%	0.12%	0.13%	0.10%	0.05%	0.14%	0.12%	0.12%	0.19%	0.22%	0.29%	0.10%	0.04%	0.45%	0.03%	0.07%	0.06%	0.05%	0.04%	0.08%	0.12%	0.16%	0.97%
2017	0.06%	0.01%	0.00%	0.00%	0.11%	0.02%	0.24%	0.01%	0.01%	0.00%	0.04%	0.04%	0.01%	0.43%	0.01%	0.01%	0.00%	0.01%	0.34%	0.01%	0.00%	0.00%	2.39%
2018	0.07%	0.10%	0.08%	0.03%	0.03%	0.04%	0.17%	0.07%	0.09%	0.00%	0.14%	0.02%	0.03%	0.62%	0.04%	0.07%	0.03%	0.04%	0.16%	0.02%	0.09%	0.14%	13.78%
2019	0.04%	0.03%	0.07%	0.00%	0.04%	0.04%	0.15%	0.05%	0.04%	0.00%	0.03%	0.00%	0.00%	0.45%	0.03%	0.02%	0.03%	0.02%	0.09%	0.00%	0.02%	0.05%	1.15%
2020	0.07%	0.10%	0.16%	0.12%	0.18%	0.15%	0.28%	0.10%	0.12%	0.16%	0.43%	0.04%	0.01%	0.56%	0.03%	0.03%	0.11%	0.01%	0.09%	0.05%	0.06%	0.13%	0.00%
MSE	0.03%	0.04%	0.03%	0.02%	0.03%	0.02%	0.06%	0.02%	0.03%	0.04%	0.06%	0.02%	0.01%	0.13%	0.01%	0.02%	0.02%	0.02%	0.06%	0.01%	0.02%	0.03%	1.98%
avg_MSE	0.12%																						

Appendix [3.2] MSE Calculating Worksheet

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
2001	45.9%	50.9%	55.9%	57.1%	57.6%	54.7%	58.9%	54.1%	53.2%	53.9%	48.7%	52.3%	55.0%	48.8%	53.4%	53.6%	49.0%	50.8%	50.4%	54.9%	53.8%	53.8%	
2002	45.9%	50.9%	55.9%	57.1%	57.6%	54.7%	58.9%	54.1%	53.2%	53.9%	48.7%	52.3%	55.0%	48.8%	53.4%	53.6%	49.0%	50.8%	50.4%	54.9%	53.8%	53.8%	50.0%
2003	46.0%	50.9%	55.9%	57.1%	57.6%	54.7%	58.9%	54.1%	53.2%	53.9%	48.7%	52.3%	55.0%	48.8%	53.4%	53.6%	49.0%	50.8%	50.4%	54.9%	53.8%	53.8%	50.0%
2004	46.0%	51.1%	55.9%	57.1%	57.6%	54.8%	58.8%	54.2%	53.2%	53.9%	48.6%	52.2%	54.9%	48.9%	53.4%	53.6%	49.0%	50.8%	50.4%	54.9%	53.8%	53.8%	50.0%
2005	46.0%	51.3%	55.9%	57.1%	57.6%	54.8%	58.6%	54.3%	53.2%	53.8%	48.4%	52.1%	54.8%	48.8%	53.5%	53.5%	49.1%	50.8%	50.4%	54.9%	53.9%	53.7%	50.8%
2006	45.9%	51.4%	55.7%	57.0%	57.5%	54.8%	58.1%	54.3%	53.2%	53.5%	48.2%	52.0%	54.6%	48.9%	53.5%	53.5%	49.1%	50.8%	50.6%	54.9%	53.9%	53.5%	50.2%
2007	45.7%	51.7%	55.5%	56.8%	57.3%	54.8%	57.7%	54.4%	53.2%	53.2%	48.0%	52.0%	54.4%	49.1%	53.5%	53.4%	49.1%	50.9%	50.6%	55.0%	53.9%	53.3%	50.3%
2008	45.7%	52.0%	55.3%	56.7%	57.1%	54.8%	57.3%	54.4%	53.2%	52.8%	47.9%	52.0%	54.3%	49.1%	53.6%	53.4%	49.1%	51.0%	50.6%	55.1%	53.7%	53.2%	51.0%
2009	46.3%	52.2%	55.1%	56.5%	56.9%	54.7%	57.0%	54.4%	53.1%	52.5%	47.8%	52.0%	54.1%	49.0%	53.7%	53.4%	49.2%	51.1%	50.8%	55.1%	53.6%	53.1%	51.6%
2010	47.0%	52.5%	55.0%	56.4%	56.7%	54.6%	56.8%	54.5%	53.1%	52.3%	47.9%	52.0%	54.0%	48.8%	53.8%	53.5%	49.5%	51.4%	51.2%	55.3%	53.6%	53.0%	52.5%
2011	47.0%	52.8%	54.9%	56.3%	56.7%	54.5%	56.6%	54.5%	53.0%	52.3%	48.1%	52.1%	54.0%	48.8%	54.0%	53.6%	49.8%	51.7%	51.6%	55.4%	53.6%	52.9%	52.2%
2012	47.3%	53.0%	54.9%	56.2%	56.6%	54.4%	56.4%	54.5%	53.0%	52.3%	48.3%	52.2%	54.0%	48.6%	54.1%	53.6%	50.0%	51.9%	51.9%	55.4%	53.7%	52.9%	52.5%
2013	47.3%	53.3%	54.9%	56.2%	56.5%	54.3%	56.3%	54.4%	53.0%	52.4%	48.5%	52.3%	54.1%	48.6%	54.3%	53.7%	50.0%	52.0%	52.3%	55.4%	53.7%	52.9%	53.0%
2014	47.3%	53.5%	54.9%	56.2%	56.3%	54.2%	56.0%	54.3%	53.0%	52.5%	48.6%	52.5%	54.1%	48.6%	54.4%	53.8%	50.0%	52.1%	52.7%	55.3%	53.8%	52.8%	51.6%
2015	47.0%	53.6%	54.9%	56.2%	56.2%	54.1%	55.8%	54.2%	53.1%	52.4%	48.6%	52.4%	54.1%	48.8%	54.3%	53.8%	50.0%	52.2%	52.8%	55.3%	53.8%	52.7%	49.7%

2016	47.3%	53.6%	54.9%	56.1%	56.0%	54.1%	55.6%	54.2%	53.1%	52.2%	48.4%	52.5%	54.1%	48.9%	54.2%	53.9%	49.9%	52.2%	53.0%	55.2%	53.9%	52.7%	49.3%
2017	48.1%	53.9%	55.1%	56.0%	55.9%	54.1%	55.3%	54.2%	53.2%	52.3%	48.2%	52.7%	54.2%	49.1%	54.2%	54.1%	50.0%	52.3%	53.1%	55.1%	54.0%	52.7%	47.6%
2018	48.9%	54.3%	55.5%	56.2%	56.1%	54.4%	55.3%	54.5%	53.6%	52.8%	48.5%	53.1%	54.4%	49.7%	54.3%	54.5%	50.2%	52.5%	53.3%	55.3%	54.3%	53.0%	45.6%
2019	49.5%	54.6%	55.8%	56.3%	55.9%	54.5%	55.0%	54.7%	53.9%	53.1%	48.6%	53.6%	54.5%	49.7%	54.4%	54.9%	50.4%	52.7%	53.0%	55.5%	54.6%	53.2%	43.0%
2020	50.2%	55.0%	56.2%	56.5%	55.9%	54.8%	55.0%	55.1%	54.4%	53.3%	48.9%	54.0%	54.6%	50.3%	54.7%	55.3%	50.6%	52.9%	53.1%	55.7%	54.9%	53.7%	44.0%
2021	51.1%	58.8%	60.7%	57.7%	57.1%	57.6%	57.9%	59.2%	58.8%	54.6%	52.3%	57.0%	55.8%	58.3%	57.4%	59.0%	53.2%	55.3%	55.7%	57.5%	58.1%	58.1%	56.8%
2022	51.3%	59.4%	61.4%	57.9%	57.3%	58.0%	58.4%	59.8%	59.5%	54.9%	52.8%	57.5%	56.0%	59.6%	57.8%	59.6%	53.6%	55.7%	56.1%	57.7%	58.7%	58.8%	58.9%

Appendix [3.3] Single Exponential Smoothing Result

	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	22	24	25	31
at	51.0%	58.2%	59.9%	57.5%	56.9%	57.1%	57.5%	58.5%	58.1%	54.4%	51.7%	56.5%	55.6%	57.0%	57.0%	58.4%	52.8%	54.9%	55.3%	57.2%	57.6%	57.4%	54.8%
bt	0.14%	0.59%	0.71%	0.19%	0.20%	0.45%	0.46%	0.65%	0.70%	0.22%	0.53%	0.48%	0.18%	1.28%	0.43%	0.59%	0.41%	0.39%	0.41%	0.28%	0.51%	0.70%	2.05%

Appendix [3.4] 2020 a_t and b_t