Analyze Crimes In Boston

Team5: Hao Cui, PinHo Wang, Tianju Zhou

Use Cases

1. Users input SQL queries and receive lists of matching records

2. The system will cluster criminal locations according to giving crime longitude and latitude

3. The system will predict the number of crimes in the next few days

4. Users input parameters and the system will search fuzzily according to the search condition

Methodology

1. Ingest data from csv files
2. Use ***Spark*** to process data (data cleaning and missing data processing)
3. Implement ***K-Means*** algorithm to cluster crime locations according to longitude and latitude
4. Use ***Holt Winter Model*** to predict crime numbers in the next 365 days based on the records from 2015 to 2018
5. Build Data Searching System with ***fuzzy matching***
6. Visualize data through ***Zeppelin*** notebook and ***Play Framework***

Data Source

Crimes in Boston

https://www.kaggle.com/AnalyzeBoston/crimes-in-boston

1. Provided by Analyze Boston

2. 17 columns (INCIDENT_NUMBER, OFFENSE_CODE, OFFENSE_CODE_GROUP, OFFENSE_DESCRIPTION, DISTRICT, REPORTING_AREA, SHOOTING, OCCURRED_ON_DATE, YEAR, MONTH, DAY_OF_WEEK, HOUR, UCR_PART, STREET, Lat, Long, Location)

3. 319073 pieces of data in the dataset.

4. The records begin in June 14, 2015 and continue to September 3, 2018.

Milestones

Data Processing:

- Deal with missing data
- Remove unnecessary data
- Build the program frame- work to analyze data

11.15

Analyze:

- Analyze data with Spark SQL
- Implement K-Means Algorithm
- Use Holt Winter Model to predict crime numbers in next few days
- Implement data search system

11.22

Data Visualization:

- Visualize data on Zeppelin with DataFrame and SparkSQL
- Build front-end web pages using Play Framework

11.29

Optimization:

- Optimize the program
- Prepare for the final presentation

12.6

Code

The program will be coded in Scala and Spark.
The UI is built with Html, CSS and JS.

Repository:
https://github.com/CSYE7200/Analyze-Crimes-Boston

Acceptance Criteria

1. All Spark SQLs should be executed within 5 seconds

select dayOfWeek, count(1) from crimes group by dayOfWeek order by count(1) desc. √ 2.0s

select offenseCodeGroup, count(1) from crimes group by offenseCodeGroup order by count(1) desc limit 15 √ 1.5s

select hour, count(1) from crimes group by hour √ 1.4s

select month, count(1) from crimes where year = 2016 or year = 2017 group by month √ 1.8s

select year, month, date, count(1) from crimes group by year, month, date order by year, month, date √ 1.5s

select * from predict √ 0.5s

Acceptance Criteria

2. The Alpha, Beta and Gamma value of Holt Winter Model should be less than 0.5

Alpha: 0.0 ✓
Beta: 0.014 ✓
Gamma: 0.538 ✗

3. The Sum of Squared Errors (SSE) of K-Means should be less than 40

SSE: 38.69 ✓

Acceptance Criteria

4. The accuracy of data search system should higher than 80%

OFFENSE_CODE_GROUP has 104 violent words, 87 (84%) of them are correct ✓

Goals

01

- Learn to use Zeppelin, Spark and Spark SQL to analyze big data
- Learn to utilize Play Framework to build simple front end pages
- Understand the basic usage of Holt Winter Model
- Gain more knowledge in K-Means and fuzzy search

02

Provide information:

- The most frequent crime type
- The most dangerous place in Boston
- The most dangerous month in a year
- The most dangerous day in a week
- The most dangerous hour in a day
- High crime rate locations in Boston
- Fuzzy search system users can use

Benefit: Tell the user where and when is the most DANGEROUS in Boston

Demo

Data Clean

Data Clean

INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPO
I182070945	619	Larceny	LARCENY ALL OTHERS	D14	
I182070943	1402	Vandalism	VANDALISM	C11	
I182070941	3410	Towed	TOWED MOTOR VEHICLE	D4	
I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	
I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	
I182070936	3820	Motor Vehicle Accident Response	M/V ACCIDENT INVOLVING PEDESTRIAN - INJURY	C11	
I182070933	724	Auto Theft	AUTO THEFT	B2	
I182070932	3301	Verbal Disputes	VERBAL DISPUTE	B2	
I182070931	301	Robbery	ROBBERY - STREET	C6	
I182070929	3301	Verbal Disputes	VERBAL DISPUTE	C11	
I182070928	3301	Verbal Disputes	VERBAL DISPUTE	C6	
I182070927	3114	Investigate Property	INVESTIGATE PROPERTY	C6	
I182070923	3108	Fire Related Reports	FIRE REPORT - HOUSE, BUILDING, ETC.	D4	
I182070922	2647	Other	THREATS TO DO BODILY HARM	B3	
I182070921	3201	Property Lost	PROPERTY - LOST	B3	
I182070920	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON		
I182070919	3301	Verbal Disputes	VERBAL DISPUTE	C11	
I182070918	3305	Assembly or Gathering Violations	DEMONSTRATIONS/RIOT	D4	
I182070917	2647	Other	THREATS TO DO BODILY HARM	B2	
I182070915	614	Larceny From Motor Vehicle	LARCENY THEFT FROM MV - NON-ACCESSORY	B2	
I182070913	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON		

Data Clean

INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPO
I182070945	619	Larceny	LARCENY ALL OTHERS	D14	
I182070943	1402	Vandalism	VANDALISM	C11	
I182070941	3410	Towed	TOWED MOTOR VEHICLE	D4	
I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	
I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	
I182070936	3820	Motor Vehicle Accident Response	M/V ACCIDENT INVOLVING PEDESTRIAN - INJURY	C11	
I182070933	724	Auto Theft	AUTO THEFT	B2	
I182070932	3301	Verbal Disputes	VERBAL DISPUTE	B2	
I182070931	301	Robbery	ROBBERY - STREET	C6	
I182070929	3301	Verbal Disputes	VERBAL DISPUTE	C11	
I182070928	3301	Verbal Disputes	VERBAL DISPUTE	C6	
I182070927	3114	Investigate Property	INVESTIGATE PROPERTY	C6	
I182070923	3108	Fire Related Reports	FIRE REPORT - HOUSE, BUILDING, ETC.	D4	
I182070922	2647	Other	THREATS TO DO BODILY HARM	B3	
I182070921	3201	Property Lost	PROPERTY - LOST	B3	
I182070920	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON		
I182070919	3301	Verbal Disputes	VERBAL DISPUTE	C11	
I182070918	3305	Assembly or Gathering Violations	DEMONSTRATIONS/RIOT	D4	
I182070917	2647	Other	THREATS TO DO BODILY HARM	B2	
I182070915	614	Larceny From Motor Vehicle	LARCENY THEFT FROM MV - NON-ACCESSORY	B2	
I182070913	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON		

Data Clean

Distinct the duplicated “Incident_number”

Drop out the row which has null value

```
scala> val data_r = spark.read.format("csv").option("header", "true").load("data_r.csv").dropDuplicates("INCIDENT_NUMBER")
data_r: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [INCIDENT_NUMBER: string, OFFENSE_CODE_GROUP: string ... 4 more fields]
```

```
scala> val joinDF = crime.as("c").join(data_r.as("d"), $"c.INCIDENT_NUMBER" === $"d.INCIDENT_NUMBER").select($"c.*")
joinDF: org.apache.spark.sql.DataFrame = [INCIDENT_NUMBER: string, OFFENSE_CODE: string ... 15 more fields]
```

Data Clean

INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA
I182070943	1402	Vandalism	VANDALISM	C11	347
I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	272
I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	421
I182070936	3820	Motor Vehicle Accident Response	M/V ACCIDENT INVOLVING PEDESTRIAN - INJURY	C11	398
I182070933	724	Auto Theft	AUTO THEFT	B2	330
I182070932	3301	Verbal Disputes	VERBAL DISPUTE	B2	584
I182070931	301	Robbery	ROBBERY - STREET	C6	177
I182070929	3301	Verbal Disputes	VERBAL DISPUTE	C11	364
I182070928	3301	Verbal Disputes	VERBAL DISPUTE	C6	913
I182070927	3114	Investigate Property	INVESTIGATE PROPERTY	C6	936
I182070923	3108	Fire Related Reports	FIRE REPORT - HOUSE, BUILDING, ETC.	D4	139
I182070922	2647	Other	THREATS TO DO BODILY HARM	B3	429
I182070919	3301	Verbal Disputes	VERBAL DISPUTE	C11	341
I182070918	3305	Assembly or Gathering Violations	DEMONSTRATIONS/RIOT	D4	130
I182070917	2647	Other	THREATS TO DO BODILY HARM	B2	901
I182070915	614	Larceny From Motor Vehicle	LARCENY THEFT FROM MV - NON-ACCESSORY	B2	181
I182070911	3801	Motor Vehicle Accident Response	M/V ACCIDENT - OTHER	A1	69
I182070909	3803	Motor Vehicle Accident Response	M/V ACCIDENT - PERSONAL INJURY	E5	550
I182070904	802	Simple Assault	ASSAULT SIMPLE - BATTERY	C11	242
I182070904	2007	Restraining Order Violations	VIOL. OF RESTRAINING ORDER W NO ARREST	C11	242
I182070903	2900	Other	VAL - VIOLATION OF AUTO LAW - OTHER	B3	463
I182070901	2907	Violations	VAL - OPERATING AFTER REV/SUSP.	B3	428
I182070900	2629	Harassment	HARASSMENT	B3	464
I182070898	802	Simple Assault	ASSAULT SIMPLE - BATTERY	C11	351
I182070895	3207	Property Found	PROPERTY - FOUND	A7	30
I182070893	614	Larceny From Motor Vehicle	LARCENY THEFT FROM MV - NON-ACCESSORY	B3	427
I182070891	3109	Police Service Incidents	SERVICE TO OTHER PD INSIDE OF MA.	E13	303
I182070890	2612	Fire Related Reports	FIRE REPORT/ALARM - FALSE	B3	432
I182070887	1402	Vandalism	VANDALISM	D4	149
I182070886	3802	Motor Vehicle Accident Response	M/V ACCIDENT - PROPERTY DAMAGE	C11	402
I182070882	3801	Motor Vehicle Accident Response	M/V ACCIDENT - OTHER	B2	901
I182070881	1402	Vandalism	VANDALISM	C11	356
I182070879	3301	Verbal Disputes	VERBAL DISPUTE	B3	441

K-means Clustering

K-Means

- Understand which location has most crime events
- Input: Latitude & Longitude; Output: K clusters
- Total number of crime events in each clusters

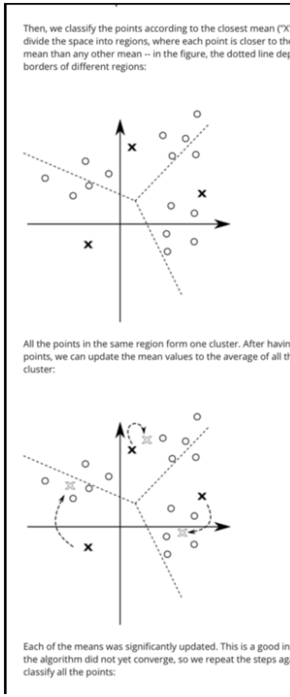
K-Means

We set K = 10

Assign 10 initial location data points as the centroids of the clusters

2 Steps:

1. Classify all data points into these clusters
2. Update 10 centroids by calculating the mean posit each clusters



Then, we classify the points according to the closest mean
divide the space into regions, where each point is closer to
mean than any other mean -- in the figure, the dotted line
borders of different regions:

All the points in the same region form one cluster. After
points, we can update the mean values to the average of all
cluster:

Each of the means was significantly updated. This is a good
the algorithm did not yet converge, so we repeat the steps
classify all the points:

K-Means

```scala
/**
 * This method takes a generic sequence of points and a generic sequence of means.
 * It returns a generic map collection, which maps each mean to the sequence of
 * points in the corresponding cluster.
 */
def classify(points: GenSeq[Point], means: GenSeq[Point]): GenMap[Point, GenSeq[Point]] = {
  val groups = points.groupBy { x => findClosest(x, means) }
  means.foldLeft(groups)((groups: GenMap[Point, GenSeq[Point]], x: Point) =>
    if (groups.contains(x)) groups
    else groups ++ GenMap[Point, GenSeq[Point]]((x, GenSeq())))
}
```

```scala
/**
 * update the new means points according to the oldmeans and classfied old means points
 */
def update(classified: GenMap[Point, GenSeq[Point]], oldMeans: GenSeq[Point]): GenSeq[Point] = {
  for (oldMean <- oldMeans) yield findAverage(classified.get(oldMean).get)
}
```

```scala
/**
 * KMeans algorithm in a tail recursive way
 * 1. get the classification of all means points and their corresponding data points
 * 2. get the updated new means sequence
 * 3. judge whether old means points and new means points satisfy converged threshold
 * 4. do the tail recursive and return the new means result
 */
@tailrec
final def kMeans(points: GenSeq[Point], means: GenSeq[Point], maxIter:Int, eta: Double): GenSeq[Point] = {
  val classification = classify(points, means)
  val newMeans = update(classification, means)
  if (!converged(eta)(means, newMeans) && maxIter >= 0) { // not converged yet and the iteration does not count to 0
    println(maxIter)
    kMeans(points, newMeans, maxIter - 1, eta)
  } else newMeans
}
```

Demo

Data Clean(part 2)

- How to judge if a crime is a violent crime?

We can firstly simplify this question to another question!

Data Clean(part 2)

- How to judge if a word is violent-related word?

WordNet?

WordNet is a Dictionary and In WordNet every word has three features

Ws = synset of the word

Wc = class word set

We = all entity in sense explanation

And use formula

$$Similarity(SW_i, SW_j) = \frac{1}{No(SWi) \times No(SWj)} \times \frac{\sum_{w_i \in \{Wsi\} \cap \{Wsj\}} Ks \times IDF(w_i)^2 + \sum_{w_i \in \{Wci\} \cap \{Wcj\}} Kc \times IDF(w_i)^2 + \sum_{w_i \in \{Wei\} \cap \{Wej\}} Ke \times IDF(w_i)^2}{\sqrt{\sum_{i \in Q_U, K \in \{Ks, Kc, Ke\}} K \times IDF(w_i)^2} \times \sqrt{\sum_{j \in Q_v, K \in \{Ks, Kc, Ke\}} K \times IDF(w_j)^2}}$$

To calculate if a word is similar with word “violent”

Data Clean(part 2)

- But it really hard to reproduce and how to estimate the threshold of similarity relationship.

Data Clean(part 2)

- How about word vertor?
- Bravo!

There is glove.6B.50d.txt from google researching

It contains 50d vectors of words(totally 400000 words)

And the vector space contain's the meaning of the words

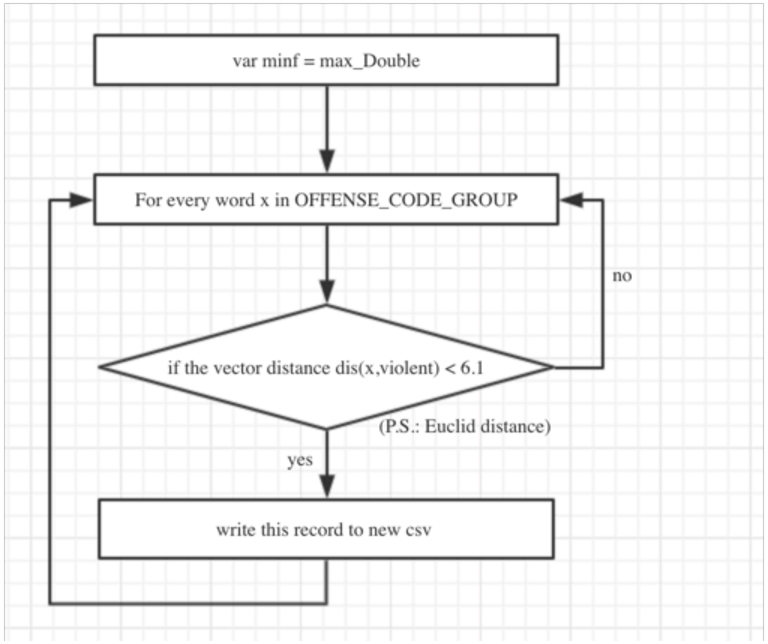
Which means more similar two words are, closer their vectors are in vector space!

Data Clean(part 2)

Some Example

counterfeiting 6.819078003384914
theft 6.07073232850806‍9
firearm 7.28408881212096
recovered 6.831614421546345‍5
harassment 4.882566870107055‍5
robbery 6.081833435610433‍5
assembly 7.00536759834778‍3
trafficking 5.591677188149008
vandalism 5.330226015822969
criminal 5.315085769987909
arrests 4.061862455092108
incidents 3.8936889207592076
involuntary 6.597775299470805

Data Clean(part 2)



Data searching System

- The building processing of PlayFrame seems boring and no need to explain in details(although it really take my a lot of time to get familiar with it)
- What I want to show is something interesting on searching processing.

Data searching System

- The work of me is to finish a work flow that takes inputs of data, street, crimes type, and returns a list of matching record.

- How to find matching record?

- Exact matching? (rigid🙅‍♂️) Fuzzy matching!

Data searching System

- Should we use any hard and complex ml algorithm to do fuzzy matching
- In these case, not we just want to handle the cases that user has wrong typing.
- We can use dynamic programming to calculate minimum edit distance of Strings.

(There is another thinking is I want to combine the knowledge of two courses , INFO 6205 and CSYE 7200)

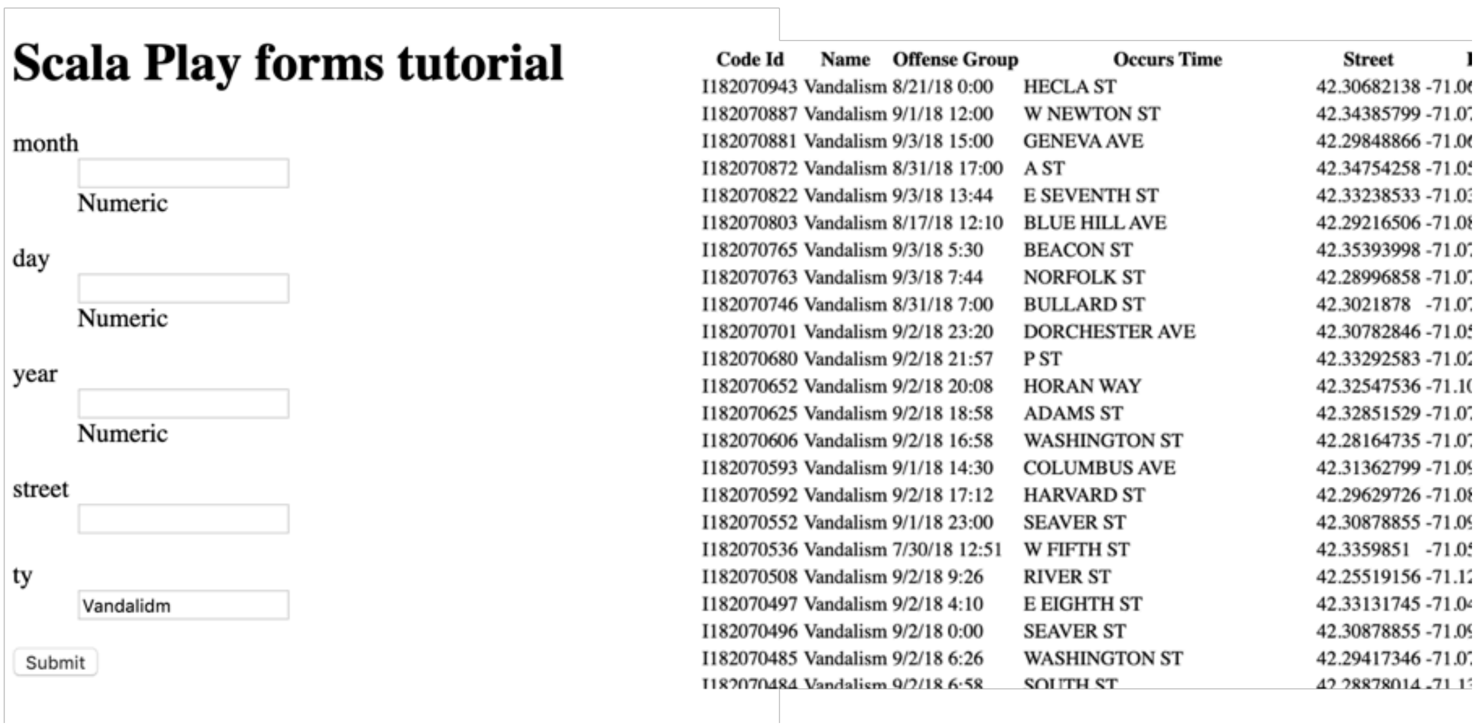
Data searching System

- Suppose there are two String s1 and s2
- State
  f(i)(j) represents the min cost when I change s1[0..i] to s2[0..j]
- Transfer equation
  f(i)(j) = min(f(i-1)(j)+1,f(i)(j-1)+1, s1[i] == s2[j]?f(i-1)(j-1):f(i-1)(j-1)+1)

Data searching System

- Imaging that In these case we have a Seq[String] sq from each record and a String s for user input
- We use min(minEditDistance(s, x)) (x <- sq) as the score of record so we can get top n records.
- How about a Seq[String] in_sq as user input?
- We use ∑min(minEditDistance(s, x)) (s <- in_sq) (x<-sq) as the score of record

Data searching System



Scala Play forms tutorial

month

Numeric

day

Numeric

year

Numeric

street

ty

Vandalidm

Submit

Code Id	Name	Offense Group	Occurs Time	Street	I
I182070943	Vandalism	8/21/18 0:00	HECLA ST	42.30682138	-71.0
I182070887	Vandalism	9/1/18 12:00	W NEWTON ST	42.34385799	-71.0
I182070881	Vandalism	9/3/18 15:00	GENEVA AVE	42.29848866	-71.0
I182070872	Vandalism	8/31/18 17:00	A ST	42.34754258	-71.0
I182070822	Vandalism	9/3/18 13:44	E SEVENTH ST	42.33238533	-71.0
I182070803	Vandalism	8/17/18 12:10	BLUE HILL AVE	42.29216506	-71.0
I182070765	Vandalism	9/3/18 5:30	BEACON ST	42.35393998	-71.0
I182070763	Vandalism	9/3/18 7:44	NORFOLK ST	42.28996858	-71.0
I182070746	Vandalism	8/31/18 7:00	BULLARD ST	42.3021878	-71.0
I182070701	Vandalism	9/2/18 23:20	DORCHESTER AVE	42.30782846	-71.0
I182070680	Vandalism	9/2/18 21:57	P ST	42.33292583	-71.0
I182070652	Vandalism	9/2/18 20:08	HORAN WAY	42.32547536	-71.1
I182070625	Vandalism	9/2/18 18:58	ADAMS ST	42.32851529	-71.0
I182070606	Vandalism	9/2/18 16:58	WASHINGTON ST	42.28164735	-71.0
I182070593	Vandalism	9/1/18 14:30	COLUMBUS AVE	42.31362799	-71.0
I182070592	Vandalism	9/2/18 17:12	HARVARD ST	42.29629726	-71.0
I182070552	Vandalism	9/1/18 23:00	SEAVER ST	42.30878855	-71.0
I182070536	Vandalism	7/30/18 12:51	W FIFTH ST	42.3359851	-71.0
I182070508	Vandalism	9/2/18 9:26	RIVER ST	42.25519156	-71.1
I182070497	Vandalism	9/2/18 4:10	E EIGHTH ST	42.33131745	-71.0
I182070496	Vandalism	9/2/18 0:00	SEAVER ST	42.30878855	-71.0
I182070485	Vandalism	9/2/18 6:26	WASHINGTON ST	42.29417346	-71.0
I182070484	Vandalism	9/2/18 6:58	SOUTH ST	42.28878014	-71.1

Thank You