DATASET

Chicago crime records

1. 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
2. These crimes may be based upon preliminary information supplied to the Police Department by the reporting parties that have not been verified.
3. 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.
4. The dataset is in CSV format, contains about 8 million records and is about 2 GB large.

UNDERSTANDING THE DATASET

Some important attributes for the better understanding of the dataset

1. **Case Number** - The Chicago Police Department's Case Number.
2. **Date and Time** - Date and time when the incident occurred.
3. **Primary Type** - The primary description of the crime e.g. Theft, Assault, Battery.
4. **Arrest** - Indicates whether an arrest was made.
5. **Domestic** - Indicates whether the incident was domestic.
6. **Location** - The location where the incident occurred with the longitude and latitudes.

USE CASES

- Finding periods with the highest and the lowest amount of crimes for every year.
- Crimes that were reduced the most during the years in every location/district in Chicago.
- Finding particular type of crime which annually influences the drop in crimes the most.
- Finding the month of the year which was the most successful in reducing the amount of drug crimes.
- Perform a k-means clustering algorithm

CHALLENGES

- Send an external file to Hadoop
- Calling the object multiple times. (Chaining the MRJob)
- Determine when the mapper reaches the end of input

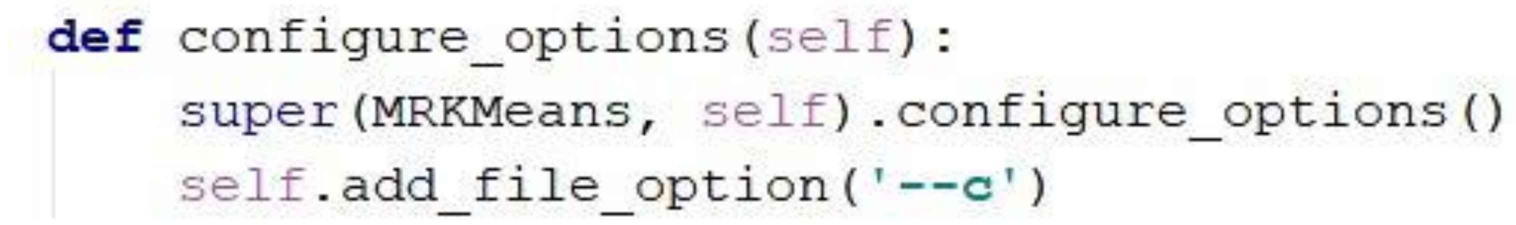
K-MEANS

- K-means algorithm:
  - Selecting k-centroids
  - For each points select the closest centroid
  - For each cluster, compute the mean point becoming the new centroid
  - Redo 2,3 until it converge

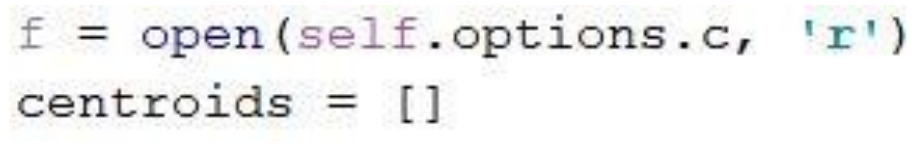
DETAILS OF IMPLEMENTATION

- Sending an external file to Hadoop

  We add a command-line option that sends an external file to Hadoop.



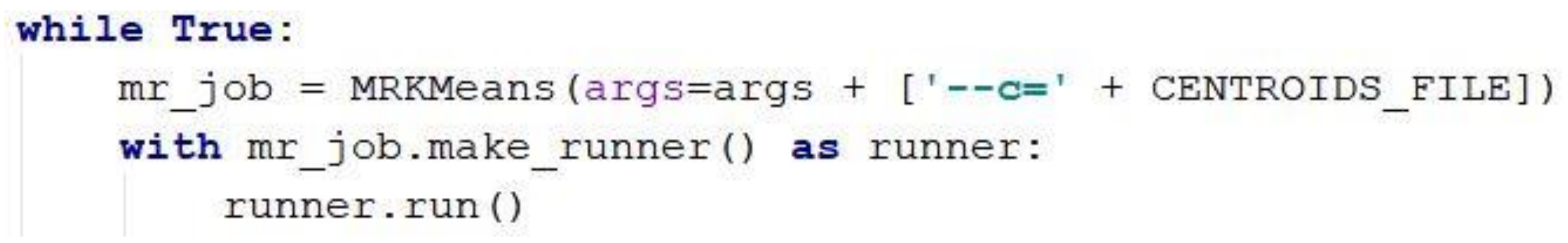
```
def configure_options(self):
    super(MRKMeans, self).configure_options()
    self.add_file_option('--c')
```



```
f = open(self.options.c, 'r')
centroids = []
```

DETAILS OF IMPLEMENTATION

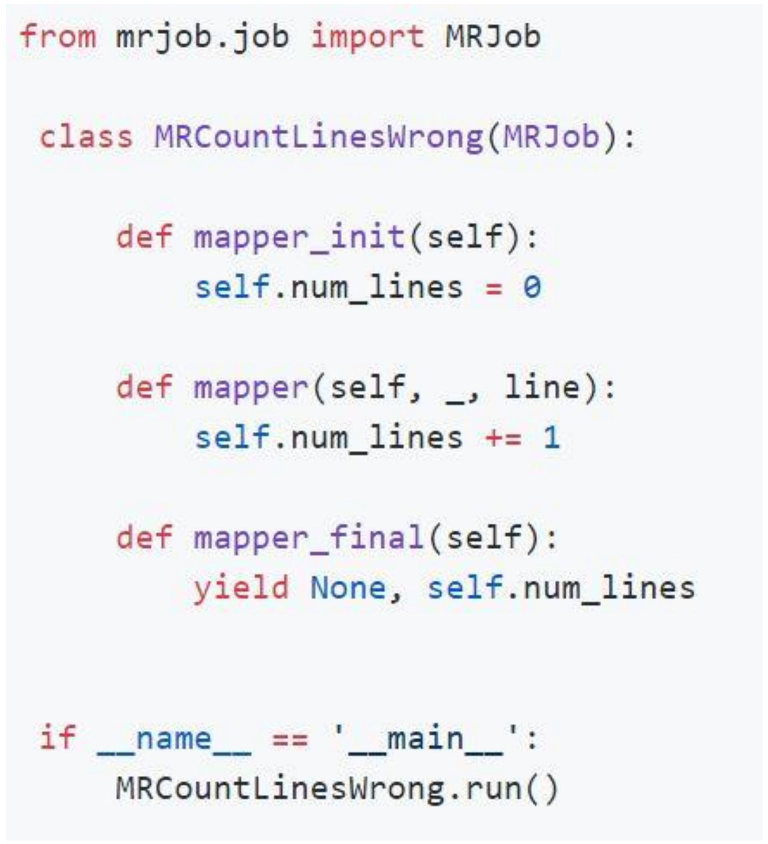
- Running a job multiple times.



```
while True:
    mr_job = MRKMeans(args=args + ['--c=' + CENTROIDS_FILE])
    with mr_job.make_runner() as runner:
        runner.run()
```

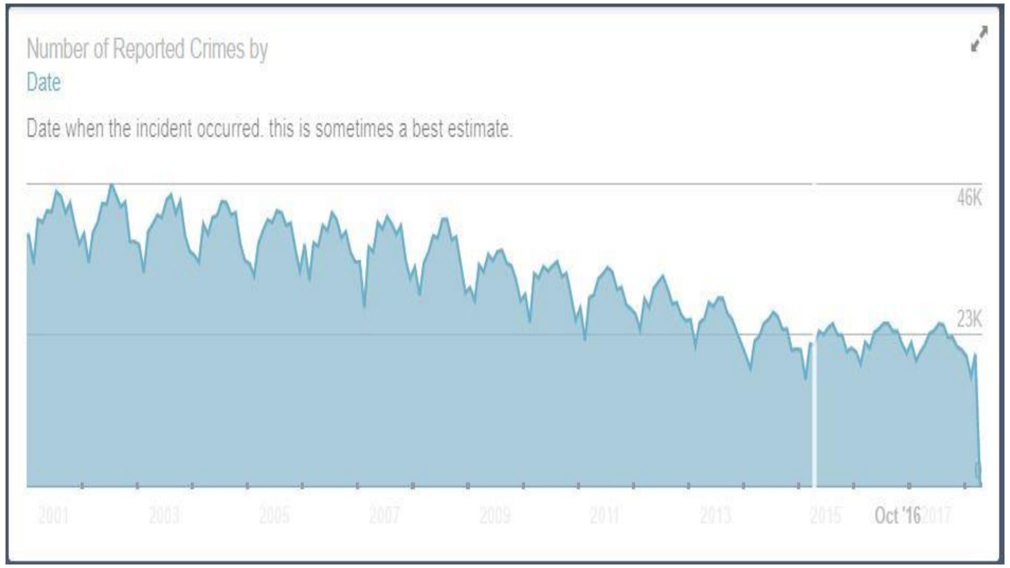
DETAILS OF IMPLEMENTATION

- Determine when the mapper reaches the end of input with mapper_final



```
from mrjob.job import MRJob

 class MRCountLinesWrong(MRJob):

     def mapper_init(self):
         self.num_lines = 0

     def mapper(self, _, line):
         self.num_lines += 1

     def mapper_final(self):
         yield None, self.num_lines


 if __name__ == '__main__':
     MRCountLinesWrong.run()
```

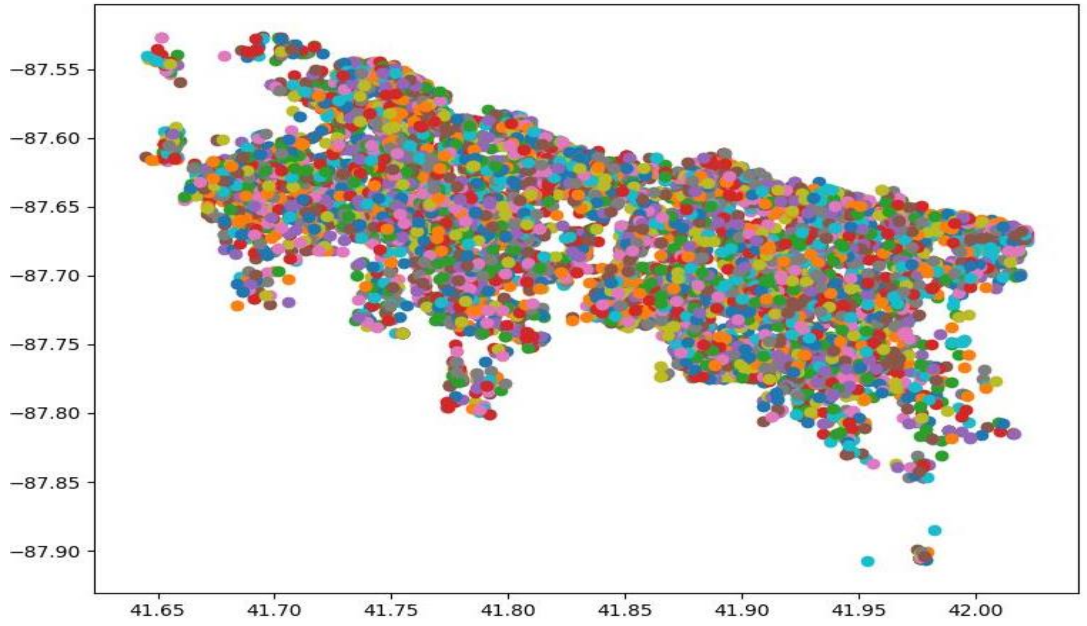
RESULTS SUMMARY



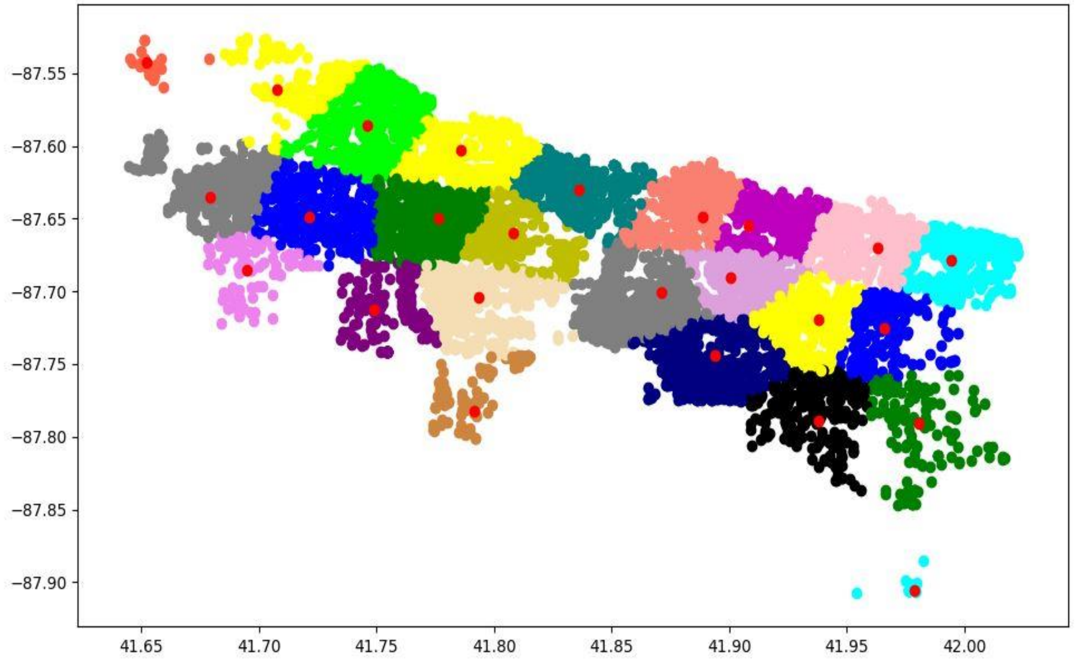
```
[2001, 2002]     ["THEFT"]
[2002, 2003]     ["BATTERY"]
[2003, 2004]     ["NARCOTICS"]
[2004, 2005]     ["THEFT"]
[2005, 2006]     ["BURGLARY"]
[2006, 2007]     ["BATTERY"]
[2008, 2009]     ["THEFT"]
[2009, 2010]     ["BATTERY"]
[2010, 2011]     ["CRIMINAL DAMAGE"]
[2011, 2012]     ["NARCOTICS"]
[2012, 2013]     ["THEFT"]
[2013, 2014]     ["BATTERY"]
[2014, 2015]     ["THEFT"]
[2015, 2016]     ["NARCOTICS"]
```

RESULT SUMMARY

Before and after clustering



−87.55
−87.60
−87.65
−87.70
−87.75
−87.80
−87.85
−87.90
41.65
41.70
41.75
41.80
41.85
41.90
41.95
42.00



−87.55
−87.60
−87.65
−87.70
−87.75
−87.80
−87.85
−87.90
41.65
41.70
41.75
41.80
41.85
41.90
41.95
42.00