**San Francisco City Crime Data Analysis**

Aditya Bhatkar [[abhatkar@uncc.edu](mailto:abhatkar@uncc.edu)]

Prakash Wagle [[pwagle@uncc.edu](mailto:pwagle@uncc.edu)]

This project aims at providing urban crime analysis. The dataset provides historic data criminal offenses in the San Francisco city. We have used K means clustering to find the clusters of dangerous city neighborhoods and Naive Bayes classifier able to advice user on the risk level of a type of crime in a neighborhood at given location and Day of the week. Both of this algorithm have been implemented using Apache Spark Framework. Our aim in the implementation of this project was to show our understanding of this Machine Learning algorithms and Spark Framework.

**Utility of the project:**

This project is aimed at aiding law enforcement officers to tackle the crime in city.

Project helps users in two ways-

1. Enables an officer to understand which crimes type she/he can expect while on duty in a city district.

Since different crimes require different counter measures out system can help officers to prepare according to the most probable issue they can face in a city district on that day.

1. Provides authorities a larger picture of crime clusters. We have created clusters of locations in the cities which can be considered dangerous.

User can give crime type and system will show clusters of locations where crime has happened.

This will help authorities to understand which areas in city need to be given more emphasis to tackle a crime type.

**Data:**

The dataset is taken from one of the problems in kaggle competitions. : <https://www.kaggle.com/c/sf-crime/data>

We initially analyzed the data set using Tableau. Data was analyzed in order figure out which Attribute can be considered as good Feature to be used in our Algorithm. Moreover we also wanted to find out whether it is possible to extract enough information to implement K means - clustering based on provided Longitude & Latitude data.

Following Data Fields were used in analysis:

Category - category of the crime incident (only in train.csv)

DayOfWeek - the day of the week

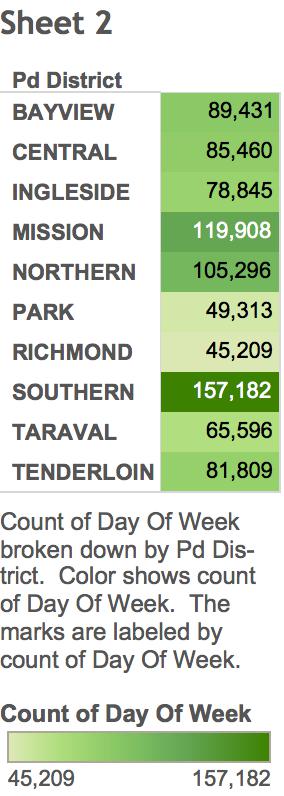
PdDistrict - name of the Police Department District

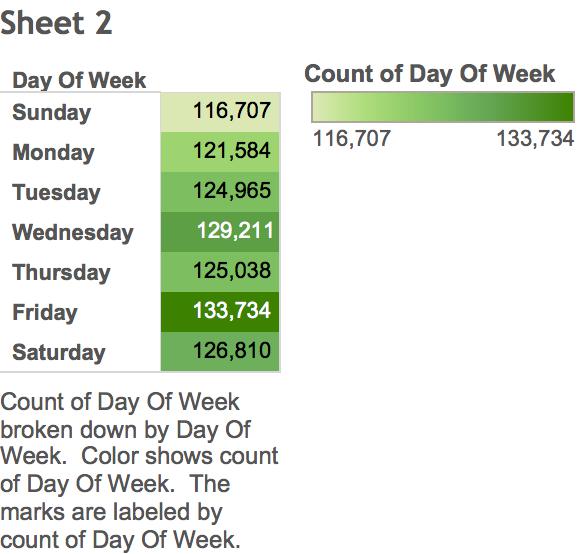
X - Longitude

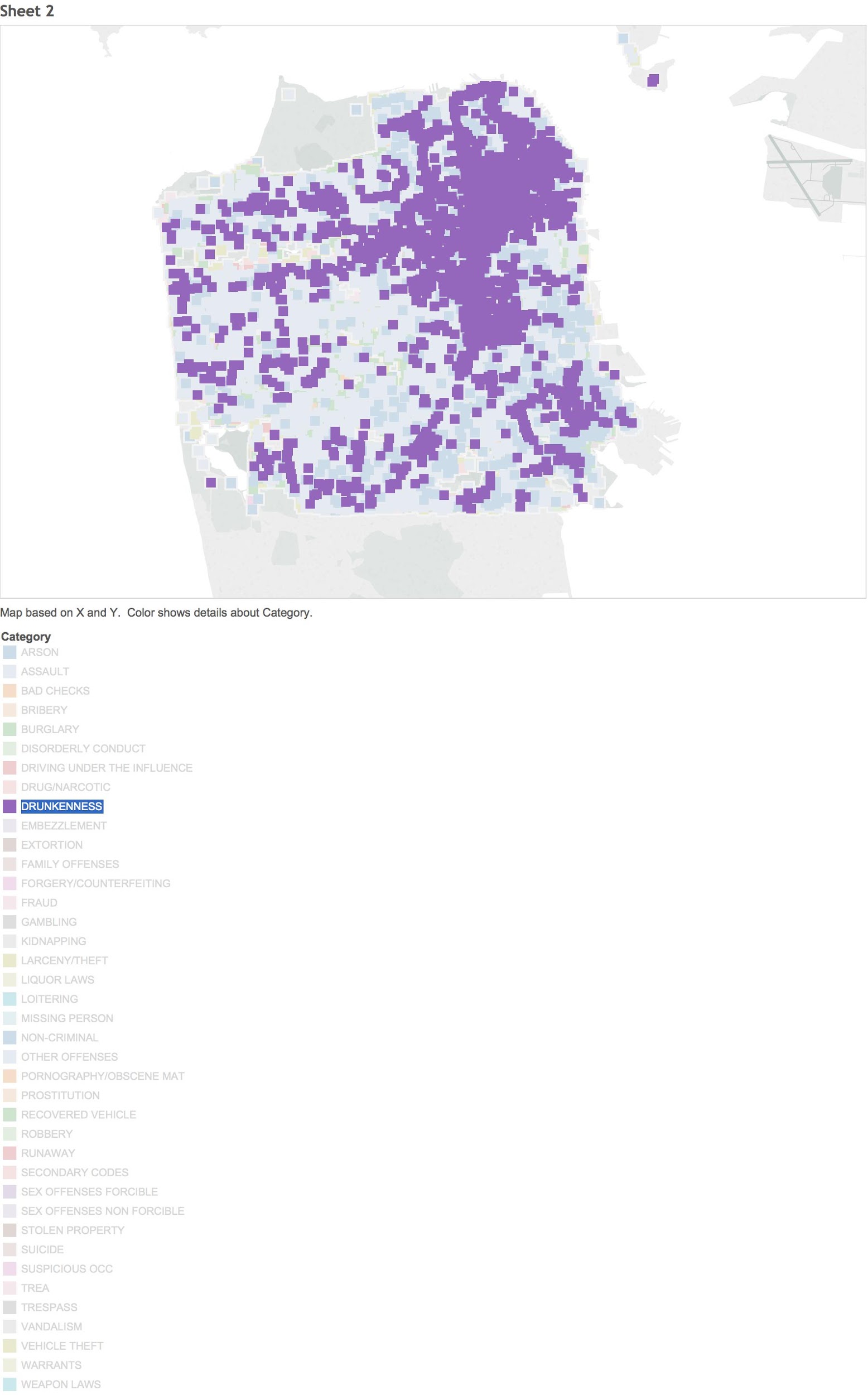
Y – Latitude

As show below we found that data has been well distributed along the location as well as day, therefore we decided to implement Naïve Bayes using Category of Crime | Day of Week | PdDistrict

We also mapped give Longitude & Latitude data on an Open Map and filtered it based on Category of Crime in order to visualize the clusters.

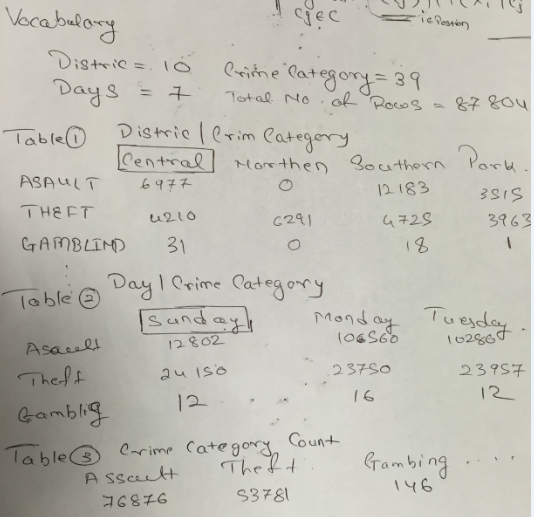






**Naïve Bayes Classifier**

For Naïve Bayes we have taken a lazy evaluation approach. Here as show below we are only going to calculate probability of based on user query. For example as shown below we will calculate probability only for “Central” and “Sunday” instead of calculating for all the Days and Location.



**Performance:**

Naïve Bayes Classifier: 878050 rows are processed | Time Taken: 60sec

K Means Clustering: 878050 rows are processed | No. of Iterations: 3 | Time Taken: 45 sec

**Software Packages:**

We in our project implementation have used Data Frame package from Data Bricks.

Data Frame enable us to organize data in form of columns which are equivalent to tables in Relational Database.

Use of this package helped us a lot in implementing Naïve Bayes Classifier.

**Goals Achieved:**

**We were able to complete one achievable goal and a ideal accomplishment goal that we had set in our project proposal.**

Achievable Goal: Clustering of dangerous city neighborhoods: K-Means clustering

Ideal Accomplishment: The system will be able to advice user on the risk level in a neighborhood at a given time and day

**Algorithms:**

1. K Means: To get centroid name if point is centroid

|  |
| --- |
| isCentroid(point, centroidArray)  compare each point in centroid Array with the given point  if given point is a centroid return the centroid name  else return "Not centroid" |

To cluster name i.e. centroid name of the given point

|  |
| --- |
| getCentroid(point, centroidArray)  call isCentroid to check if given point is a centroid  if point is not centroid  find the distance of the point from every centroid  get the centroid where distance is minimum  return centroid name  else  return centroid name from isCentroid function |

To calculate distance

|  |
| --- |
| getDistance(point, centroid)  calculate euclidean distance between the point and centroid |

To create clusters

|  |
| --- |
| kmeans(dataFileName, kValue, numberOfIterations, crimeCategory)  get latitudes and longitudes from data file using data frame  create array of latitudes and longitudes  select random K points as centroids from the data  name the centroids as C1, C2, C3,.... Ck  combine the latidues and longitudes to form a RDD of points  for given numberOfIterations  getCentroid(point, centriodArray) for each point  clusters=map centroid name as key and point co-ordinates as key for each point  reduce clusters RDD by key to get all centroids  in clusters RDD we have all points with their respective centroids as key  to print centroids and number points in them  filter clusters RDD by centroid names |

Distributed computations and storage: Distributed approach provides speed and fault tolerance.

We have used below features in implementing K means

The prominent key value pair is: <”clusterName”, (latidtude, longitude)>

1. Data Frames: Data file is stored as data frames hence fetching data is fast.
2. RDDs: All data structures: longitudes, latitudes, clusters, centroids are RDDs
3. Map: Map function is applied on RDDs to

* Form key value pairs of cluster names and points
* Combine latitudes and longitudes into singe RDD.
* Create key value pairs for cluster names and their centroids

4. Reduce: Reduce is applied to collect all points in a given cluster and create new centroid.

5. Filter: To get points only if given cluster name.

1. Naïve Bayes:

|  |
| --- |
| load data in data frame from the data file  get number of crimes for each category as CattegoriesAndCount  get number of crimes for given day, pd district and their respective category as QueryCategoryAndCount  for each row in  CattegoriesAndCount  initialize probability as 0  lookup in the query RDD by category name to get the respective count  get nk  calculate probability as nk/n and store along the category  sort list by probabilities  print highest 3 categories |

Distributed computations and storage:

Distributed approach provides speed and fault tolerance.

We have used below features in implementing Naïve Bayes

The prominent key value pair is:

<”CategoryName”, count of incidents>

1. Data Frames: Data file is stored as data frames hence fetching data is fast.

2. RDDs: All data structures: longitudes, latitudes, clusters, centroids are RDDs

3. Map: Map function is applied on RDDs to Map Crime category to count if incidents

1. Look up:

This features efficiently searches required element in RDD.

By using key value feature of map.

Provide the key returns the respective value.

**Observation & Challenges:**

* Initially in our proposal we intend to use Logistic Regression to predict the risk level of type of crime that would occur given Day of Week and Location, but we fail to implement as we found that Logistic Regression can only be used for binary classification. Hence we switched to Naïve Bayes Classifier for the implementation of our proposal.
* Our input file was an in csv format and data present in some of our attributes had comma inside it, this created a challenge while reading data in Spark.
* One of the coolest feature we used in our Project was that of PairRDD function called lookup(). This function allows us to find list of value related to a key without iterating over a RDD.

**Work Division:**

Data preprocessing, conceptualization, debugging was done by both of us.

Implementation of the program was done in following way.

Naïve Bayes: Prakash Wagle

K Means Clustering: Aditya Bhatkar

**Conclusion:**

We have successfully implemented our project proposal. This project has increased our understanding of the Machine Algorithm we used. It also helped us in getting more familiar with Apache Spark Framework and the thought process of implementing such algorithms in Map Reduce framework. It was possible to implement this algorithm on such huge amount of data without using a lot of resources because of Spark Framework.

**References:**

Following important references that we used to implement our project

http://spark.apache.org/docs/latest/quick-start.html

http://spark.apache.org/docs/latest/sql-programming-guide.html

<http://spark-packages.org/package/databricks/spark-csv>

<https://en.wikipedia.org/wiki/Naive_Bayes_classifier>

<https://en.wikipedia.org/wiki/K-means_clustering>

https://www.kaggle.com/c/sf-crime