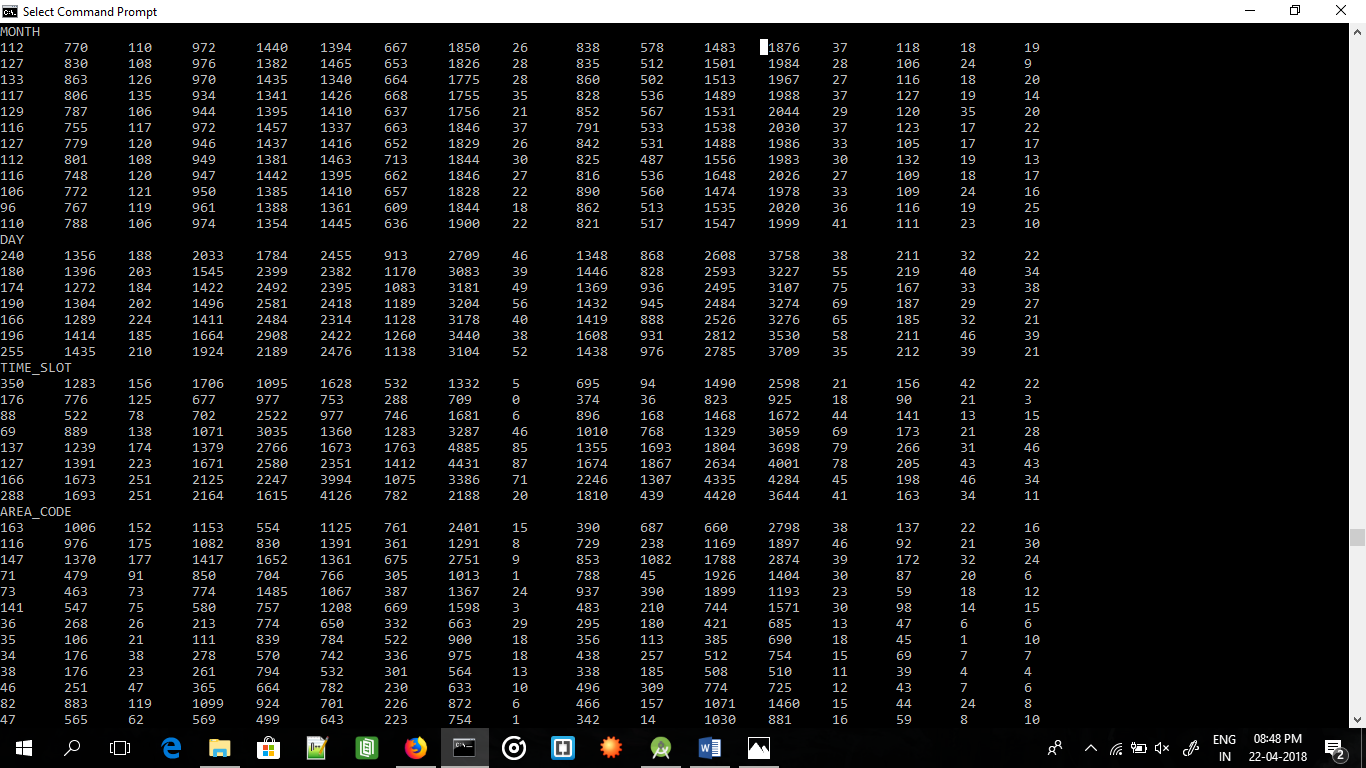
Naïve Bayesian Networks Code

import java.util.\*;  
import java.io.\*;  
public class BayesianNetwork  
{  
 public static void DisplayArrayList(ArrayList<String> al)  
 {  
 for(int i = 0; i < al.size();i++)  
 { System.*out*.println(al.get(i)); }  
 }  
 public static void DisplayMatrix(int[][] arr,int m,int n,String text)  
 { System.*out*.println(text);  
 for(int i=0;i<m;i++)  
 { for(int j=0;j<n;j++)  
 { System.*out*.print(arr[i][j]+"\t"); }  
 System.*out*.println();  
 }  
 }  
  
 public static void DisplayProb(ArrayList<String> al1,int[] count\_crime,int totalNumData)  
 { for(int i = 0; i < al1.size();i++)  
 {  
 System.*out*.println(al1.get(i)+"\t\t:"+count\_crime[i]+" /"+totalNumData);  
 }  
 }  
  
 public static void main(String[] args) throws IOException  
 {  
 //System.out.println(args[0]);  
 //System.out.println(args[1]);  
 //System.out.println(args[2]);  
  
 ArrayList<String> DistinctMonth = new ArrayList<String>();  
 ArrayList<String> DistinctDay = new ArrayList<String>();  
 ArrayList<String> DistinctAreaCode = new ArrayList<String>();  
 ArrayList<String> DistinctTimeSlot = new ArrayList<String>();  
 ArrayList<String> DistinctCrime = new ArrayList<String>();  
  
 DistinctMonth.add("January"); DistinctMonth.add("February"); DistinctMonth.add("March"); DistinctMonth.add("April");  
 DistinctMonth.add("May"); DistinctMonth.add("June"); DistinctMonth.add("July"); DistinctMonth.add("August");  
 DistinctMonth.add("September"); DistinctMonth.add("October"); DistinctMonth.add("November"); DistinctMonth.add("December");  
  
 DistinctDay.add("Sunday"); DistinctDay.add("Monday"); DistinctDay.add("Tuesday"); DistinctDay.add("Wednesday");  
 DistinctDay.add("Thursday"); DistinctDay.add("Friday"); DistinctDay.add("Saturday");  
  
 DistinctCrime.add("RAPE"); DistinctCrime.add("ROBBERY");  
 DistinctCrime.add("ATTEMPTED ROBBERY"); DistinctCrime.add("ASSAULT WITH DEADLY WEAPON");  
 DistinctCrime.add("BURGLARY"); DistinctCrime.add("VEHICLE BURGLARY");  
 DistinctCrime.add("THEFT (GRAND)"); DistinctCrime.add("PICKPOCKETING");  
 DistinctCrime.add("SHOPLIFTING (GRAND)"); DistinctCrime.add("THEFT FROM VEHICLE (PETTY)");  
 DistinctCrime.add("SHOPLIFTING (PETTY)"); DistinctCrime.add("STOLEN VEHICLE");  
 DistinctCrime.add("ASSAULT"); DistinctCrime.add("INDECENT EXPOSURE");  
 DistinctCrime.add("BATTERY WITH SEXUAL CONTACT"); DistinctCrime.add("KIDNAPPING");  
 DistinctCrime.add("EXTORTION");  
  
 for(int i=1; i <= 21; i++) { DistinctAreaCode.add(i+""); }  
  
 for(int i=1; i <= 8; i++) { DistinctTimeSlot.add("T"+i); }  
   
 /\*  
 // REMOVE TO VIEW THE DATA ENTERED  
   
 DisplayArrayList(DistinctMonth);  
 DisplayArrayList(DistinctDay);  
 DisplayArrayList(DistinctAreaCode);  
 DisplayArrayList(DistinctTimeSlot);  
 DisplayArrayList(DistinctCrime);  
 \*/  
 int[][] prob\_month = new int[12][17];  
 int[][] prob\_day = new int[7][17];  
 int[][] prob\_areacode = new int[21][17];  
 int[][] prob\_timeslot = new int[8][17];  
 int[] count\_crime = new int[17];  
 int totalNumData = 0;  
  
 BufferedReader br = new BufferedReader(new FileReader(args[0]));  
 String input = "";  
 while(true)  
 { input = br.readLine();  
 if(input.equals("stop"))  
 break;  
 //TO VIEW THE DATA FROM FILE  
 //System.out.println(input);  
 StringTokenizer st = new StringTokenizer(input,",");  
 String dayINP = st.nextToken();  
 String monthINP = st.nextToken();  
 String timeslotINP = st.nextToken();  
 String areacodeINP = st.nextToken();  
 String crimeINP = st.nextToken();  
  
 prob\_day[DistinctDay.indexOf(dayINP)][DistinctCrime.indexOf(crimeINP)]++;  
 prob\_month[DistinctMonth.indexOf(monthINP)][DistinctCrime.indexOf(crimeINP)]++;  
 prob\_areacode[DistinctAreaCode.indexOf(areacodeINP)][DistinctCrime.indexOf(crimeINP)]++;  
 prob\_timeslot[DistinctTimeSlot.indexOf(timeslotINP)][DistinctCrime.indexOf(crimeINP)]++;  
 count\_crime[DistinctCrime.indexOf(crimeINP)]++;  
 totalNumData++;  
 }  
  
 System.*out*.println("DATA ENTERED");  
 //DisplayMatrix(prob\_month,12,17,"MONTH");  
 //DisplayMatrix(prob\_day,7,17,"DAY");  
 //DisplayMatrix(prob\_timeslot,8,17,"TIME\_SLOT");  
 //DisplayMatrix(prob\_areacode,21,17,"AREA\_CODE");  
 //DisplayProb(DistinctCrime,count\_crime,totalNumData);  
  
  
 Scanner in2 = new Scanner(System.*in*);  
 String InputData = in2.nextLine();  
 StringTokenizer st = new StringTokenizer(InputData,",");  
 String dayINP = st.nextToken();  
 String monthINP = st.nextToken();  
 String timeslotINP = st.nextToken();  
 String areacodeINP = st.nextToken();  
  
 double[] allprobs = new double[17];  
  
 for(int i=0; i < DistinctCrime.size();i++)  
 {  
 int index = DistinctCrime.indexOf(DistinctCrime.get(i));  
  
 double a = (double)prob\_day[DistinctDay.indexOf(dayINP)][index] / (double)count\_crime[DistinctCrime.indexOf(DistinctCrime.get(i))];  
 double b = (double)prob\_month[DistinctMonth.indexOf(monthINP)][index] / (double)count\_crime[DistinctCrime.indexOf(DistinctCrime.get(i))];  
 double c = (double)prob\_areacode[DistinctAreaCode.indexOf(areacodeINP)][index] / (double)count\_crime[DistinctCrime.indexOf(DistinctCrime.get(i))];  
 double d = (double)prob\_timeslot[DistinctTimeSlot.indexOf(timeslotINP)][index] / (double)count\_crime[DistinctCrime.indexOf(DistinctCrime.get(i))];  
  
 double e = (double)count\_crime[index] / (double)totalNumData;  
  
 double p\_crime = a\*b\*c\*d\*e;  
  
 allprobs[i] = p\_crime;  
  
 //System.out.println(a+" "+b+" "+c+" "+d+" "+e);  
  
 System.*out*.println(DistinctCrime.get(i)+" : "+p\_crime);  
 }  
  
 double max = allprobs[0];  
 int valr = -1;  
  
 for(int i =0; i < 17; i++)  
 { if(allprobs[i] >= max)  
 {  
 max = allprobs[i];  
 valr = i;  
 }  
 }  
 System.*out*.print("\nPREDICTION ");  
 System.*out*.println(DistinctCrime.get(valr)+" "+allprobs[valr]);  
 }  
}

BAYESIAN NETWORK OUTPUT

# Conditional probability Tables ( can be viewed in better detail in report )

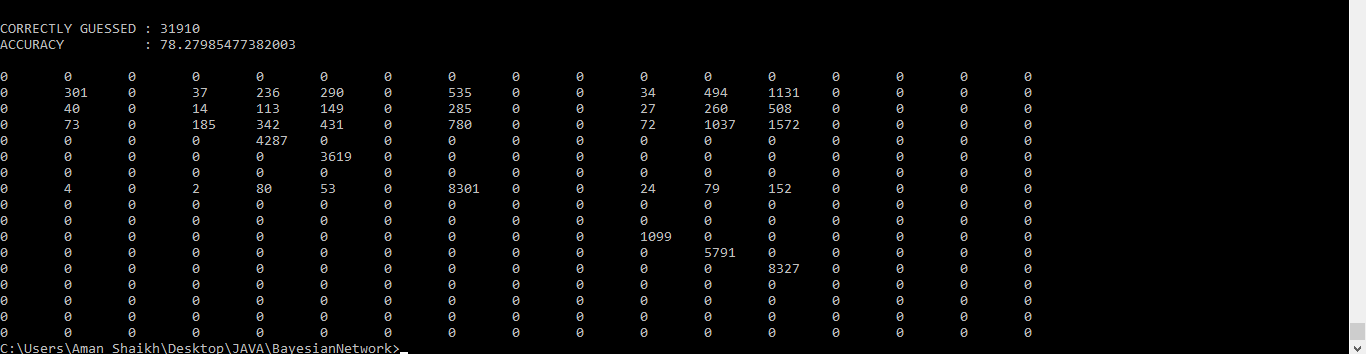


EXAMPLE OF Output, Input and all Crime probability (Very less as 148,000 entries in total in learning set)

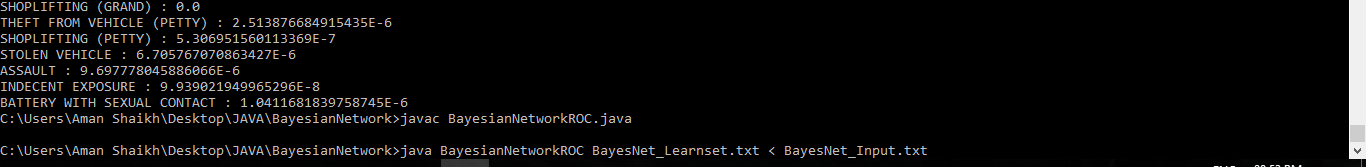
View Entire output in BayesianNet\_Output.txt



ROC Data and Accuracy



How to Run



Javac BayesianNetworkROC.java

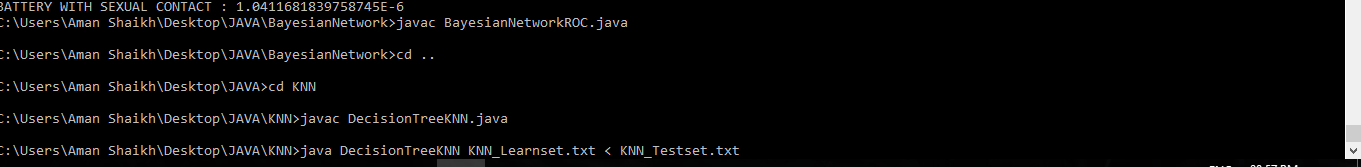
Java BayesianNetworkROC BayesNet\_Learnset.txt < BayesNet\_Input.txt

Knn with filter

The input is filtered similar to how Decision Tree would reduce the DataSet. Then Knn Classifier is used to find the Euclidean distances between the remaining data and input data and then the most frequent class determined by the k distances ( in our case the top 5 ) are considered.

import java.util.\*;  
 import java.io.\*;  
public class DecisionTreeKNN  
{  
 public static ArrayList <Integer> indexOfAll (String str, ArrayList list)  
 {  
 ArrayList<Integer> indexList = new ArrayList<Integer>();  
 for (int i = 0; i < list.size(); i++)  
 if(str.equals(list.get(i)))  
 indexList.add(i);  
 return indexList;  
 }  
  
 public static double EuclideanDistance(Double a, Double b, Double c, Double d)  
 { return Math.*sqrt*(Math.*pow*(a-b,2) + Math.*pow*(c-d,2)); }  
  
 public static void main(String[] args) throws IOException  
 {  
  
 ArrayList<String> LearnSetOne = new ArrayList<String>();  
 ArrayList<String> LearnSetCrm = new ArrayList<String>();  
 ArrayList<Double> LearnSetLat = new ArrayList<Double>();  
 ArrayList<Double> LearnSetLng = new ArrayList<Double>();  
  
 BufferedReader br = new BufferedReader(new FileReader(args[0]));  
 String input = "";  
 int totalNumData = 0;  
  
 while(true)  
 { input = br.readLine();  
 if(input.equals("stop"))  
 break;  
 //TO VIEW THE DATA FROM FILE  
 //System.out.println(input);  
 StringTokenizer st = new StringTokenizer(input,",");  
 String dayINP = st.nextToken();  
 String monthINP = st.nextToken();  
 String timeslotINP = st.nextToken();  
 String areacodeINP = st.nextToken();  
 String crimeINP = st.nextToken();  
 Double latitudeINP = Double.*parseDouble*(st.nextToken());  
 Double longitudINP = Double.*parseDouble*(st.nextToken());  
 String DecisionTreeText = dayINP+","+monthINP+","+timeslotINP+","+areacodeINP;  
  
 LearnSetOne.add(DecisionTreeText);  
 LearnSetCrm.add(crimeINP);  
 LearnSetLat.add(latitudeINP);  
 LearnSetLng.add(longitudINP);  
  
 totalNumData++;  
 }  
  
 //System.out.println("DATA ENTERED");  
  
 String InputData;  
 Scanner in2 = new Scanner(System.*in*);  
  
 int correct = 0;  
 int totz = 0;  
  
 while(true)  
 {  
 totz++;  
 InputData = in2.nextLine();  
 if(InputData.equals("stop"))  
 break;  
  
 StringTokenizer st = new StringTokenizer(InputData,",");  
 String dayINP = st.nextToken();  
 String monthINP = st.nextToken();  
 String timeslotINP = st.nextToken();  
 String areacodeINP = st.nextToken();  
 String crimeINP = st.nextToken();  
 Double latitudeINP = Double.*parseDouble*(st.nextToken());  
 Double longitudINP = Double.*parseDouble*(st.nextToken());  
  
 String DecisionTreeText = dayINP+","+monthINP+","+timeslotINP+","+areacodeINP;  
  
 ArrayList <Integer> indi = *indexOfAll*(DecisionTreeText,LearnSetOne);  
  
 int KNNsize = 0;  
  
 if(indi.size() >= 3)  
 KNNsize = 3;  
 else  
 KNNsize = indi.size();  
  
 //System.out.println(indi.size());  
  
 HashMap<Double, String> hmap = new HashMap<Double, String>();  
  
  
 for(int i = 0; i < indi.size(); i++)  
 { int index = indi.get(i);  
 //System.out.println(LearnSetOne.get(index));  
 double e = *EuclideanDistance*(latitudeINP,LearnSetLat.get(index),longitudINP,LearnSetLng.get(index));  
 //System.out.println(e +" "+ LearnSetCrm.get(index));  
 hmap.put(e,LearnSetCrm.get(index));  
 }  
  
 Map<Double, String> map = new TreeMap<Double, String>(hmap);  
 Set set2 = map.entrySet();  
 Iterator iterator2 = set2.iterator();  
  
 //REMOVE TO VIEW ALL K POSSIBLE CRIMES  
  
  
 System.*out*.println("K possible Crimes :");  
 while(iterator2.hasNext())  
 { Map.Entry me2 = (Map.Entry)iterator2.next();  
 System.*out*.print(me2.getKey() + ": ");  
 System.*out*.println(me2.getValue());  
 }  
 Set set3 = map.entrySet();  
 Iterator iterator3 = set3.iterator();  
 Map.Entry me3 = (Map.Entry)iterator3.next();  
  
 String prediction = me3.getValue()+"";  
  
 System.*out*.print("\nACTUAL : "+crimeINP);  
 System.*out*.println("\nPREDICTION : "+me3.getValue()+" @ "+me3.getKey());  
 if(crimeINP.equals(prediction)) correct++;  
  
 System.*out*.println();  
 }  
 System.*out*.println("CORRECTLY GUESSED : "+correct);  
 System.*out*.println("ACCURACY : "+((double)correct/totz)\*100);  
  
 }  
}  
/\*  
FIRST ARGS : TrainingSet  
\*/

How to Run

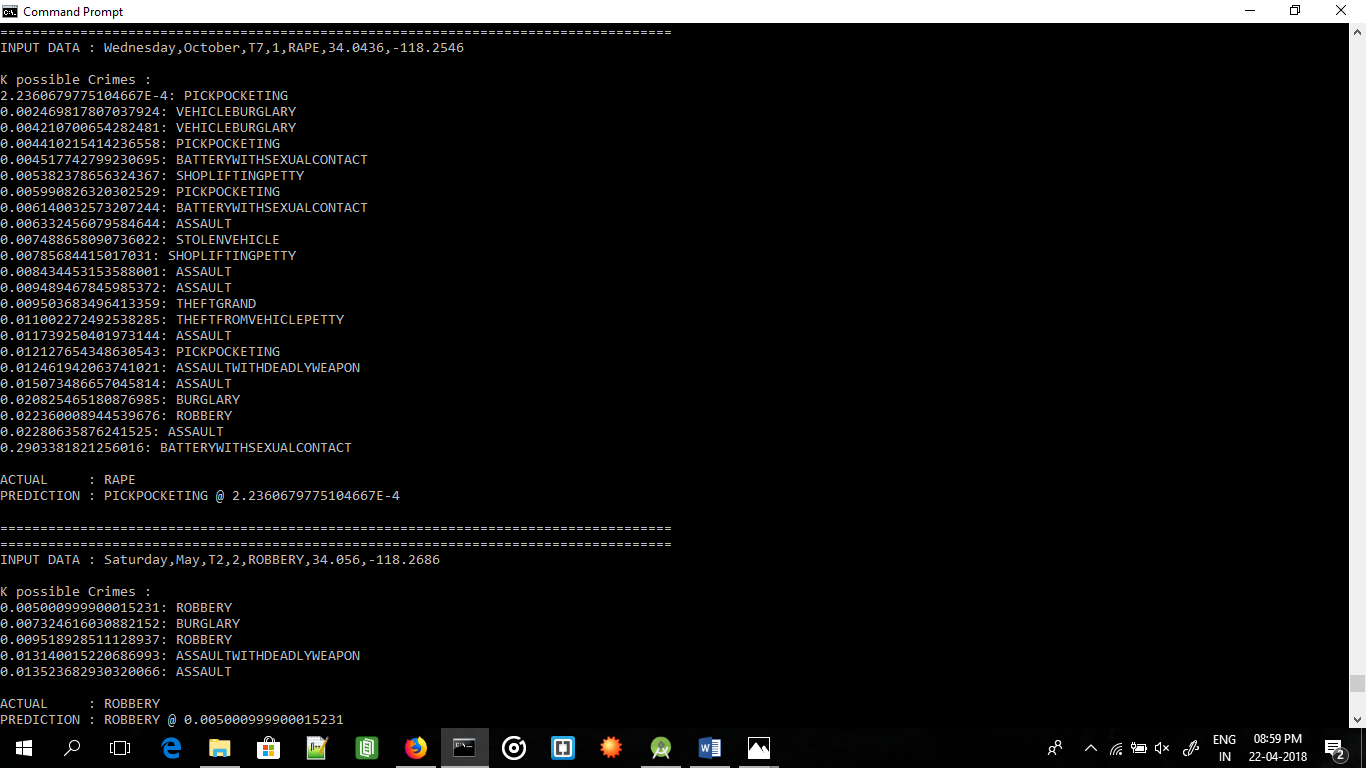


Javac DecisionTreeKNN.java

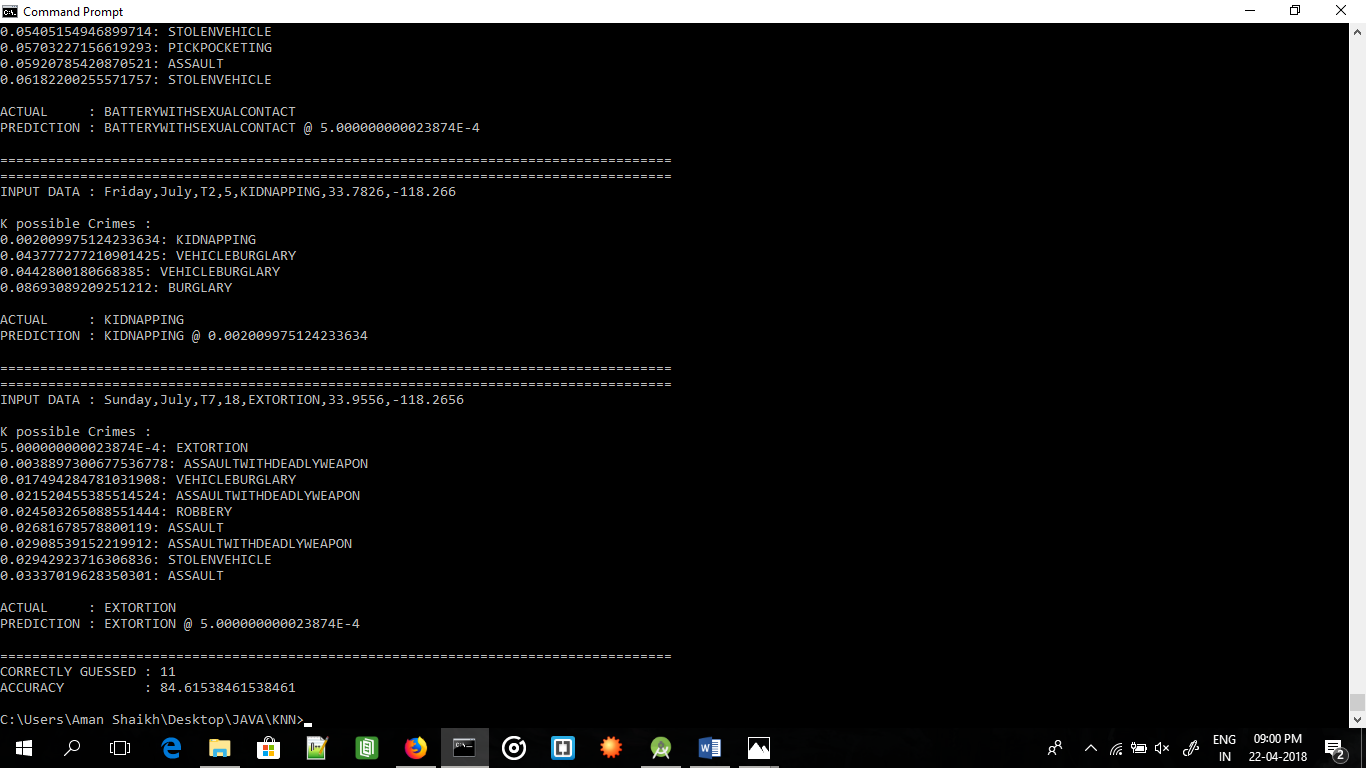
Java DecisionTreeKNN KNN\_Learnset.txt < KNN\_Testset.txt

OUTPUT :

Input and all possible crimes with their Euclidean distances in ascending order

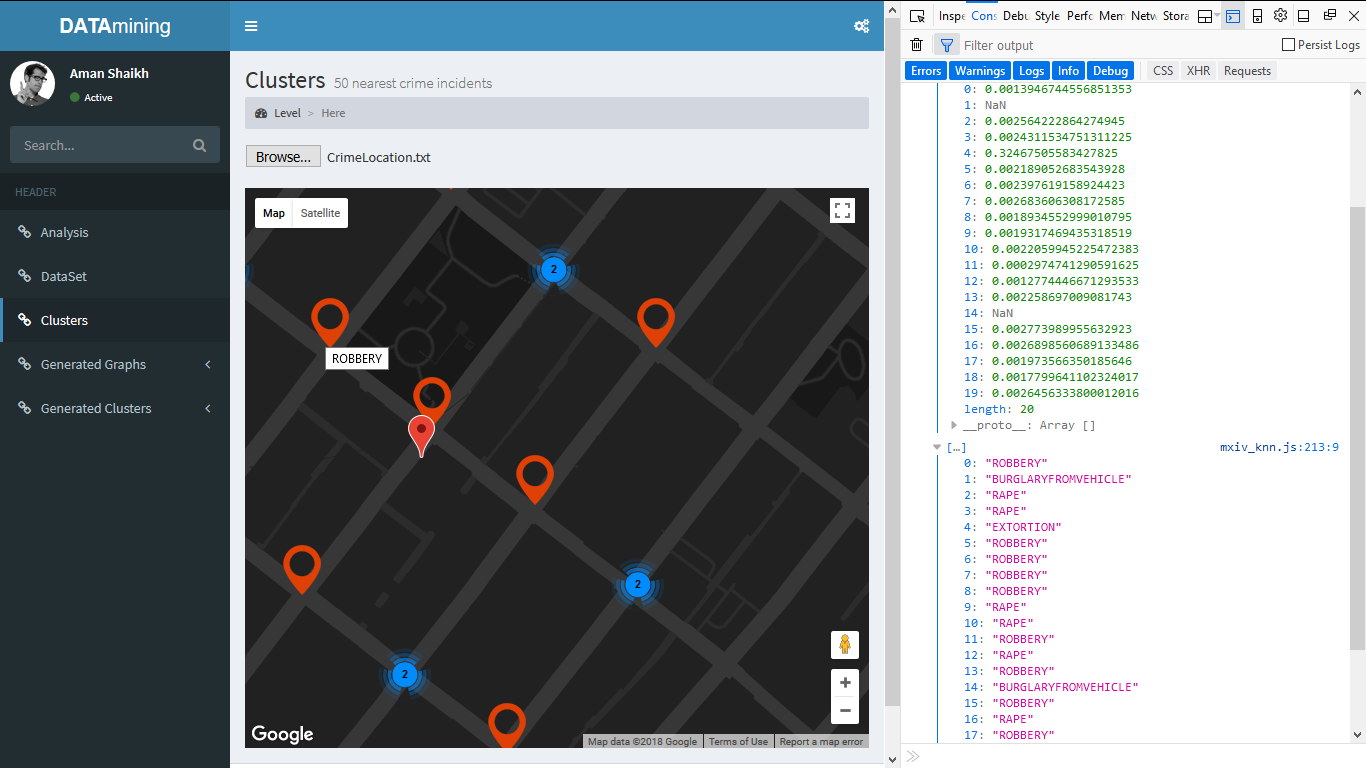
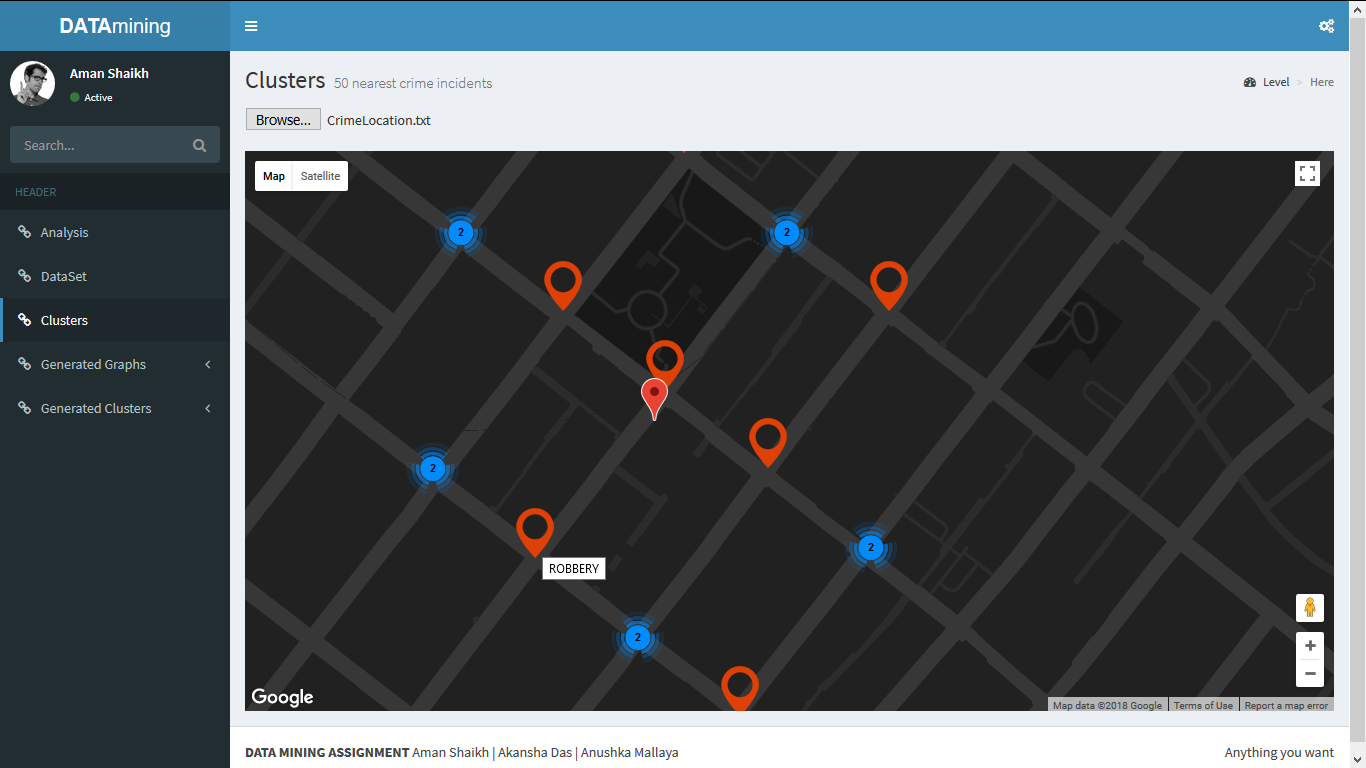


ACCURACY



Dynamic KNN Algorithm in javascript

To generate 20 nearest crimes to the position user clicked



Code ( javascript file) (see demo video for implementation)

var text;

var labelz = [];

var locations = [

{lat: 34.0438, lng: -118.2547},

{lat: 34.0141, lng: -118.2915},

{lat: 34.0309, lng: -118.3002},

{lat: 34, lng: -118.3308},

{lat: 34.0109, lng: -118.3023},

{lat: 34.0242, lng: -118.2172},

{lat: 34.0794, lng: -118.1974},

{lat: 33.7333, lng: -118.2879},

{lat: 34.0584, lng: -118.4247},

{lat: 34.0532, lng: -118.4952},

]

function initMap()

{

var map = new google.maps.Map(document.getElementById('map'),

{

zoom: 3,

center: {lat: 34.0438, lng: -118.2547},

disableDoubleClickZoom: true,

styles:

[{ "elementType": "geometry",

"stylers": [{ "color": "#212121" }]

},

{ "elementType": "labels.icon",

"stylers": [{ "visibility": "off" }]

},

{ "elementType": "labels.text.fill",

"stylers": [{ "color": "#757575" }]

},

{ "elementType": "labels.text.stroke",

"stylers": [{ "color": "#212121" }]

},

{ "featureType": "administrative",

"elementType": "geometry",

"stylers": [{ "color": "#757575" }]

},

{ "featureType": "administrative.country",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#9e9e9e" }]

},

{ "featureType": "administrative.land\_parcel",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "administrative.locality",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#bdbdbd" }]

},

{ "featureType": "administrative.neighborhood",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "poi",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#757575" }]

},

{ "featureType": "poi.business",

"stylers": [{ "visibility": "off" }]

},

{

"featureType": "poi.park",

"elementType": "geometry",

"stylers": [{ "color": "#181818" }]

},

{ "featureType": "poi.park",

"elementType": "labels.text",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "poi.park",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#616161" }]

},

{ "featureType": "poi.park",

"elementType": "labels.text.stroke",

"stylers": [{ "color": "#1b1b1b" }]

},

{ "featureType": "road",

"elementType": "geometry.fill",

"stylers": [{ "color": "#2c2c2c" }]

},

{ "featureType": "road",

"elementType": "labels",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "road",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#8a8a8a" }]

},

{ "featureType": "road.arterial",

"elementType": "geometry",

"stylers": [{ "color": "#373737" }]

},

{ "featureType": "road.highway",

"elementType": "geometry",

"stylers": [{ "color": "#3c3c3c" }]

},

{ "featureType": "road.highway.controlled\_access",

"elementType": "geometry",

"stylers": [{ "color": "#4e4e4e" }]

},

{ "featureType": "road.local",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#616161" }]

},

{

"featureType": "transit",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#757575" }]

},

{

"featureType": "water",

"elementType": "geometry",

"stylers": [{ "color": "#000000" }]

},

{

"featureType": "water",

"elementType": "labels.text",

"stylers": [{ "visibility": "off" }]

},

{

"featureType": "water",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#3d3d3d" }]

}]

});

// Create an array of alphabetical characters used to label the markers.

var labels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

// Add some markers to the map.

// Note: The code uses the JavaScript Array.prototype.map() method to

// create an array of markers based on a given "locations" array.

// The map() method here has nothing to do with the Google Maps API.

/\*

var markers = locations.map(function(location, i)

{ return new google.maps.Marker

({

position: location,

label: labelz[i],

icon: 'Pictures/PICKPOCKETING.png'

});

});

// Add a marker clusterer to manage the markers.

var markerCluster = new MarkerClusterer(map, markers,

{imagePath: 'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m'});

\*/

console.log(locations);

var MarkerLat;

var MarkerLng;

// Create new marker on double click event on the map

google.maps.event.addListener(map,'dblclick',function(event)

{ var marker = new google.maps.Marker

({

position: event.latLng,

map: map,

title: event.latLng.lat()+', '+event.latLng.lng()

});

MarkerLat = event.latLng.lat();

MarkerLng = event.latLng.lng();

console.log(MarkerLat);

console.log(MarkerLng);

var hello = text.split('\r\n');

var knn = [];

var k = 20; //KNN VARIABLE

var N = hello.length;

for(var i = 0; i < k; i++)

knn[i]=999;

for(var i = 0; i < k; i++)

console.log(knn[i]); //[NaN, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999, 999]

var llt;

var llg;

for(var i=0; i<N; i++)

{

//console.log(hello[i]);

var omg = hello[i].split(',');

llt = parseFloat(omg[1]);//console.log(llt);

llg = parseFloat(omg[2]);

var distance = Math.sqrt(Math.pow(llt-MarkerLat,2) + Math.pow(llg-MarkerLng,2));

//console.log(distance);

//var index = -1;

var index = 0;

var max = knn[0];

for(var j = 1; j < k; j++)

{ if(knn[j] > max)

{ index = j;

max = knn[j];

}

}

if(knn.indexOf(distance) == -1)

{ knn[index] = distance;

labelz[index] = omg[0];

//console.log(llt);

var pos = new google.maps.LatLng(llt,llg)

//locations[index] = "{lat: "+omg[1]+", lng: "+omg[2]+"}";

locations[index] = pos;

}

}

console.log(knn);

console.log(labelz);

console.log(locations);

var markers = locations.map(function(location, i)

{ return new google.maps.Marker

({

position: location,

title: labelz[i],

icon: 'Pictures/user.png'

});

});

var markerCluster = new MarkerClusterer(map, markers,

{imagePath: 'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m'});

});

}

var openFile = function(event)

{

var input = event.target;

var reader = new FileReader();

reader.onload = function()

{

text = reader.result;

console.log("data loaded");

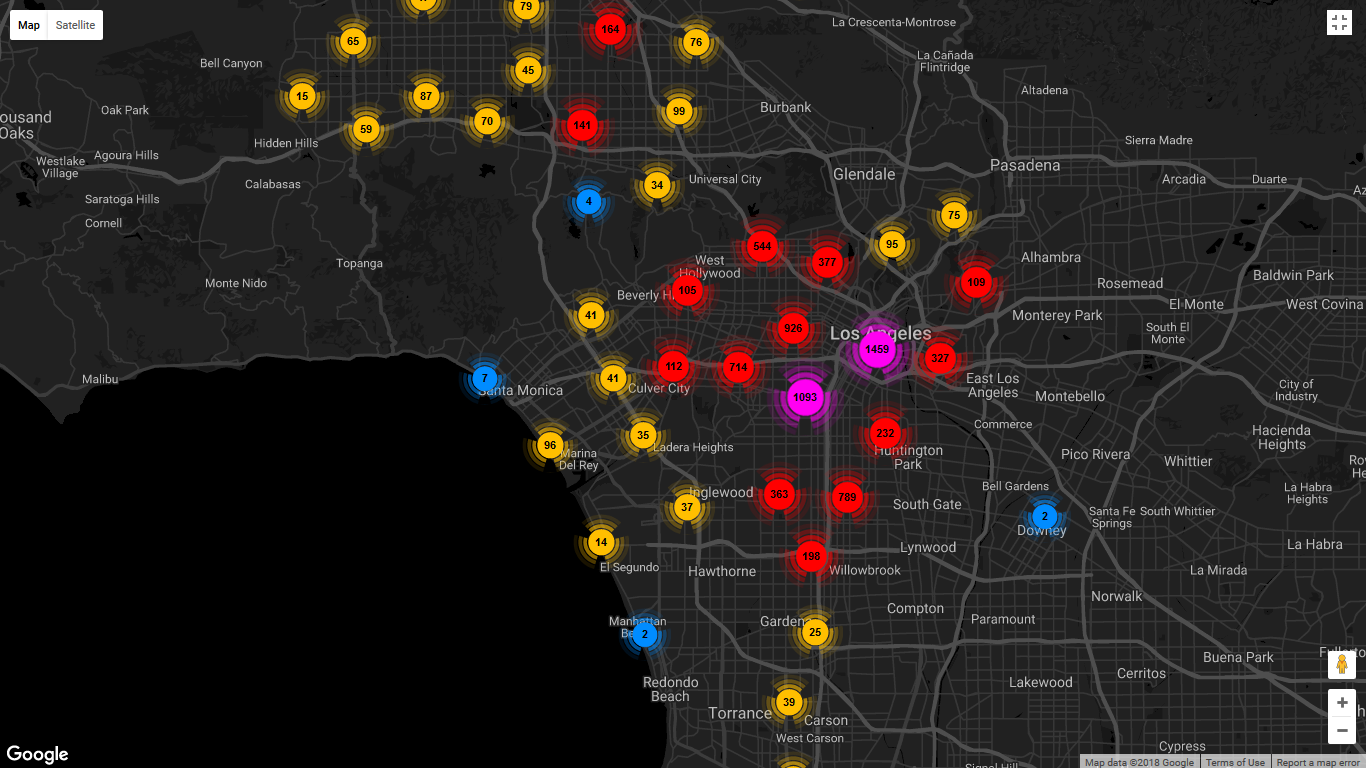
};

reader.readAsText(input.files[0]);

};

DENSITY BASED SPATIAL CLUSTERING OF APPLICATION WITH NOISE (DBSCAN)

DBSCAN was performed with Hierarchical feature which computes Clusters Dynamically according to “ZOOM” level.



CODE (Javascript file) (see demo )

function initMap()

{

var map = new google.maps.Map(document.getElementById('map'),

{

zoom: 3,

center: {lat: 34.0438, lng: -118.2547},

disableDoubleClickZoom: true,

styles:

[{ "elementType": "geometry",

"stylers": [{ "color": "#212121" }]

},

{ "elementType": "labels.icon",

"stylers": [{ "visibility": "off" }]

},

{ "elementType": "labels.text.fill",

"stylers": [{ "color": "#757575" }]

},

{ "elementType": "labels.text.stroke",

"stylers": [{ "color": "#212121" }]

},

{ "featureType": "administrative",

"elementType": "geometry",

"stylers": [{ "color": "#757575" }]

},

{ "featureType": "administrative.country",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#9e9e9e" }]

},

{ "featureType": "administrative.land\_parcel",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "administrative.locality",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#bdbdbd" }]

},

{ "featureType": "administrative.neighborhood",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "poi",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#757575" }]

},

{ "featureType": "poi.business",

"stylers": [{ "visibility": "off" }]

},

{

"featureType": "poi.park",

"elementType": "geometry",

"stylers": [{ "color": "#181818" }]

},

{ "featureType": "poi.park",

"elementType": "labels.text",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "poi.park",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#616161" }]

},

{ "featureType": "poi.park",

"elementType": "labels.text.stroke",

"stylers": [{ "color": "#1b1b1b" }]

},

{ "featureType": "road",

"elementType": "geometry.fill",

"stylers": [{ "color": "#2c2c2c" }]

},

{ "featureType": "road",

"elementType": "labels",

"stylers": [{ "visibility": "off" }]

},

{ "featureType": "road",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#8a8a8a" }]

},

{ "featureType": "road.arterial",

"elementType": "geometry",

"stylers": [{ "color": "#373737" }]

},

{ "featureType": "road.highway",

"elementType": "geometry",

"stylers": [{ "color": "#3c3c3c" }]

},

{ "featureType": "road.highway.controlled\_access",

"elementType": "geometry",

"stylers": [{ "color": "#4e4e4e" }]

},

{ "featureType": "road.local",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#616161" }]

},

{

"featureType": "transit",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#757575" }]

},

{

"featureType": "water",

"elementType": "geometry",

"stylers": [{ "color": "#000000" }]

},

{

"featureType": "water",

"elementType": "labels.text",

"stylers": [{ "visibility": "off" }]

},

{

"featureType": "water",

"elementType": "labels.text.fill",

"stylers": [{ "color": "#3d3d3d" }]

}]

});

// Create an array of alphabetical characters used to label the markers.

var labels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

// Add some markers to the map.

// Note: The code uses the JavaScript Array.prototype.map() method to

// create an array of markers based on a given "locations" array.

// The map() method here has nothing to do with the Google Maps API.

var markers = locations.map(function(location, i)

{ return new google.maps.Marker

({

position: location,

title: 'ROBBERY',

icon: 'Pictures/user.png'

});

});

// Add a marker clusterer to manage the markers.

var markerCluster = new MarkerClusterer(map, markers,

{imagePath: 'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m'});

var MarkerLat;

var MarkerLng;

// Create new marker on double click event on the map

google.maps.event.addListener(map,'dblclick',function(event)

{ var marker = new google.maps.Marker

({

position: event.latLng,

map: map,

title: event.latLng.lat()+', '+event.latLng.lng()

});

MarkerLat = event.latLng.lat();

MarkerLng = event.latLng.lng();

console.log(MarkerLat);

console.log(MarkerLng);

});

}

var locations = [

{lat: 34.0759, lng: -118.2553}, {lat: 34.2048, lng: -118.4488}, {lat: 34.2048, lng: -118.4488}, {lat: 34.2048, lng: -118.4488}, {lat: 33.9727, lng: -118.2914}, {lat: 34.1016, lng: -118.331}, {lat: 34.0468, lng: -118.2052}, {lat: 34.2012, lng: -118.3878}, {lat: 34.0105, lng: -118.3398}, {lat: 34.0105, lng: -118.3398},

{lat: 34.0105, lng: -118.3398}, {lat: 34.2198, lng: -118.4671}, {lat: 34.0759, lng: -118.2553}, {lat: 34.0155, lng: -118.2975}, {lat: 34.0155, lng: -118.2975}, {lat: 34.2612, lng: -118.4288}, {lat: 34.2217, lng: -118.4072}, {lat: 34.0933, lng: -118.3025}, {lat: 34.0933, lng: -118.3025}, {lat: 34.2687, lng: -118.4279},

{lat: 34.0052, lng: -118.4281}, {lat: 34.1479, lng: -118.3877}, {lat: 34.0182, lng: -118.3078}, {lat: 34.0472, lng: -118.309}, {lat: 34.0228, lng: -118.3089}, {lat: 34.0897, lng: -118.2893}, {lat: 34.0598, lng: -118.2916}, {lat: 34.0562, lng: -118.194}, {lat: 34.2233, lng: -118.3964}, {lat: 34.0557, lng: -118.2488},

{lat: 34.0037, lng: -118.3089}, {lat: 33.9507, lng: -118.2461}, {lat: 34.0438, lng: -118.2841}, {lat: 34.2067, lng: -118.3682}, {lat: 34.0749, lng: -118.2868}, {lat: 33.9951, lng: -118.3002}, {lat: 34.254, lng: -118.5155}, {lat: 34.1425, lng: -118.3643}, {lat: 33.9792, lng: -118.2937}, {lat: 34.0989, lng: -118.2739},

{lat: 33.9543, lng: -118.3003}, {lat: 34.186, lng: -118.6522}, {lat: 34.2487, lng: -118.4098}, {lat: 34.2318, lng: -118.4503}, {lat: 33.7723, lng: -118.271}, {lat: 33.9638, lng: -118.2695}, {lat: 34.0073, lng: -118.2485}, {lat: 34.0188, lng: -118.1973}, {lat: 34.2588, lng: -118.4321}, {lat: 33.7493, lng: -118.2876},

{lat: 33.982, lng: -118.2614}, {lat: 34.066, lng: -118.2116}, {lat: 34.0328, lng: -118.2864}, {lat: 34.0342, lng: -118.2816}, {lat: 34.0559, lng: -118.28}, {lat: 34.0428, lng: -118.2461}, {lat: 34.1259, lng: -118.1864}, {lat: 33.9933, lng: -118.2827}, {lat: 33.9933, lng: -118.2827}, {lat: 33.7451, lng: -118.2901},

{lat: 33.9579, lng: -118.4174}, {lat: 34.2157, lng: -118.5118}, {lat: 34.0479, lng: -118.3447}, {lat: 34.2085, lng: -118.4749}, {lat: 33.9897, lng: -118.2915}, {lat: 34.033, lng: -118.2125}, {lat: 34.0468, lng: -118.3273}, {lat: 34.1522, lng: -118.3527}, {lat: 34.0462, lng: -118.2802}, {lat: 34.0478, lng: -118.3267},

{lat: 34.2048, lng: -118.3921}, {lat: 33.933, lng: -118.2695}, {lat: 33.7906, lng: -118.2739}, {lat: 33.9747, lng: -118.2827}, {lat: 34.0529, lng: -118.2329}, {lat: 34.0659, lng: -118.1939}, {lat: 33.7415, lng: -118.2923}, {lat: 33.7787, lng: -118.2584}, {lat: 34.0443, lng: -118.211}, {lat: 34.0073, lng: -118.298},

{lat: 34.0596, lng: -118.2749}, {lat: 34.0907, lng: -118.3332}, {lat: 34.0363, lng: -118.2672}, {lat: 34.047, lng: -118.2755}, {lat: 33.7315, lng: -118.2857}, {lat: 34.0467, lng: -118.2485}, {lat: 34.0998, lng: -118.3502}, {lat: 34.0501, lng: -118.2592}, {lat: 34.0453, lng: -118.2651}, {lat: 34.0757, lng: -118.1711},

{lat: 34.01, lng: -118.2915}, {lat: 34.0249, lng: -118.369}, {lat: 34.0182, lng: -118.3002}, {lat: 34.057, lng: -118.2437}, {lat: 34.071, lng: -118.2302}, {lat: 33.9092, lng: -118.2914}, {lat: 34.0108, lng: -118.3171}, {lat: 34.0577, lng: -118.3053}, {lat: 34.057, lng: -118.2779}, {lat: 34.0583, lng: -118.2616},

{lat: 34.0998, lng: -118.3267}, {lat: 34.0146, lng: -118.3414}, {lat: 34.095, lng: -118.3068}, {lat: 34.0167, lng: -118.4202}, {lat: 34.0395, lng: -118.2405}, {lat: 34.0282, lng: -118.276}, {lat: 34.2247, lng: -118.4677}, {lat: 34.0155, lng: -118.3002}, {lat: 34.0411, lng: -118.3517}, {lat: 34.0386, lng: -118.2694},

{lat: 34.1392, lng: -118.1981}, {lat: 34.0512, lng: -118.2093}, {lat: 33.9456, lng: -118.2739}, {lat: 34.0271, lng: -118.4106}, {lat: 34.0763, lng: -118.2994}, {lat: 34.2031, lng: -118.4313}, {lat: 34.0036, lng: -118.3319}, {lat: 34.001, lng: -118.2783}, {lat: 34.0418, lng: -118.309}, {lat: 33.9932, lng: -118.3177},

{lat: 33.96, lng: -118.3161}, {lat: 34.1675, lng: -118.6042}, {lat: 34.0467, lng: -118.247}, {lat: 33.9383, lng: -118.263}, {lat: 34.0444, lng: -118.2628}, {lat: 33.7484, lng: -118.2933}, {lat: 34.0328, lng: -118.2903}, {lat: 34.0111, lng: -118.2774}, {lat: 34.1576, lng: -118.379}, {lat: 34.015, lng: -118.3536},

{lat: 34.015, lng: -118.3536}, {lat: 34.2595, lng: -118.4167}, {lat: 34.1375, lng: -118.2173}, {lat: 34.0293, lng: -118.241}, {lat: 34.0836, lng: -118.2892}, {lat: 34.0742, lng: -118.1731}, {lat: 34.1197, lng: -118.1982}, {lat: 33.9933, lng: -118.3046}, {lat: 33.9798, lng: -118.4656}, {lat: 34.0574, lng: -118.2675},

{lat: 34.0276, lng: -118.3248}, {lat: 34.2067, lng: -118.3878}, {lat: 34.0544, lng: -118.2767}, {lat: 34.03, lng: -118.2801}, {lat: 34.0508, lng: -118.2078}, {lat: 34.0572, lng: -118.2481}, {lat: 34.0814, lng: -118.2962}, {lat: 33.97, lng: -118.309}, {lat: 33.9598, lng: -118.4167}, {lat: 33.983, lng: -118.3265},

{lat: 33.983, lng: -118.3265}, {lat: 33.9601, lng: -118.2782}, {lat: 34.0116, lng: -118.3412}, {lat: 34.0609, lng: -118.2052}, {lat: 33.9311, lng: -118.2652}, {lat: 34.0216, lng: -118.3556}, {lat: 34.057, lng: -118.2118}, {lat: 33.7828, lng: -118.264}, {lat: 33.9782, lng: -118.2827}, {lat: 34.2919, lng: -118.4582},

{lat: 34.0384, lng: -118.2571}, {lat: 33.7907, lng: -118.2729}, {lat: 34.0436, lng: -118.3249}, {lat: 33.7206, lng: -118.2907}, {lat: 34.0918, lng: -118.3092}, {lat: 34.0873, lng: -118.1716}, {lat: 34.0167, lng: -118.2092}, {lat: 34.2152, lng: -118.4488}, {lat: 34.0639, lng: -118.2721}, {lat: 33.9601, lng: -118.2739},

{lat: 33.7899, lng: -118.2975}, {lat: 33.9446, lng: -118.2826}, {lat: 34.0255, lng: -118.2959}, {lat: 33.9601, lng: -118.2761}, {lat: 34.0409, lng: -118.3177}, {lat: 34.0294, lng: -118.2671}, {lat: 34.0234, lng: -118.2177}, {lat: 33.8018, lng: -118.3034}, {lat: 34.1016, lng: -118.3297}, {lat: 34.0327, lng: -118.3041},

{lat: 33.9292, lng: -118.2652}, {lat: 34.2658, lng: -118.5014}, {lat: 34.0526, lng: -118.2111}, {lat: 34.0896, lng: -118.3092}, {lat: 34.0599, lng: -118.2368}, {lat: 34.1939, lng: -118.4531}, {lat: 34.0132, lng: -118.2945}, {lat: 33.7478, lng: -118.2879}, {lat: 33.8818, lng: -118.2851}, {lat: 34.2377, lng: -118.4159},

{lat: 33.9834, lng: -118.3134}, {lat: 33.9587, lng: -118.2827}, {lat: 34.0421, lng: -118.2405}, {lat: 34.1946, lng: -118.5821}, {lat: 34.098, lng: -118.3287}, {lat: 34.2012, lng: -118.4749}, {lat: 34.2012, lng: -118.4859}, {lat: 33.7289, lng: -118.2846}, {lat: 34.0981, lng: -118.3158}, {lat: 34.0343, lng: -118.4478},

{lat: 34.059, lng: -118.2725}, {lat: 34.1685, lng: -118.6058}, {lat: 34.0256, lng: -118.312}, {lat: 33.7801, lng: -118.2608}, {lat: 34.0099, lng: -118.3089}, {lat: 34.1016, lng: -118.3361}, {lat: 33.9942, lng: -118.2783}, {lat: 34.1907, lng: -118.4465}, {lat: 34.1907, lng: -118.4465}, {lat: 34.1681, lng: -118.3765},

{lat: 34.0562, lng: -118.2673}, {lat: 34.053, lng: -118.2648}, {lat: 34.1793, lng: -118.5905}, {lat: 34.0417, lng: -118.2443}, {lat: 34.2355, lng: -118.5885}, {lat: 34.0633, lng: -118.2738}, {lat: 34.194, lng: -118.4181}, {lat: 34.0088, lng: -118.327}, {lat: 34.2439, lng: -118.4164}, {lat: 33.7924, lng: -118.2511},

{lat: 33.9329, lng: -118.2826}, {lat: 34.2285, lng: -118.4271}, {lat: 34.0368, lng: -118.3328}, {lat: 34.0305, lng: -118.2865}, {lat: 33.96, lng: -118.309}, {lat: 34.0596, lng: -118.2749}, {lat: 33.7916, lng: -118.2562}, {lat: 33.9727, lng: -118.2893}, {lat: 33.9968, lng: -118.4207}, {lat: 34.0419, lng: -118.2485},

{lat: 34.0597, lng: -118.2795}, {lat: 34.2482, lng: -118.5886}, {lat: 33.951, lng: -118.2827}, {lat: 33.9798, lng: -118.2827}, {lat: 34.2275, lng: -118.399}, {lat: 33.9747, lng: -118.2783}, {lat: 33.9456, lng: -118.2826}, {lat: 34.1667, lng: -118.5908}, {lat: 33.9534, lng: -118.2475}, {lat: 33.9874, lng: -118.2652},

{lat: 34.2011, lng: -118.5863}, {lat: 34.0481, lng: -118.2507}, {lat: 34.0742, lng: -118.2519}, {lat: 34.0359, lng: -118.2344}, {lat: 34.0739, lng: -118.1952}, {lat: 34.0926, lng: -118.3223}, {lat: 34.091, lng: -118.2277}, {lat: 34.0591, lng: -118.2412}, {lat: 34.0437, lng: -118.2098}, {lat: 34.0437, lng: -118.2098},

{lat: 33.9833, lng: -118.3003},

]

Utility Functions

Functions to get data for intermediate purposes and graphs.

GetLocation is to get coordinates of crimes which is supplied to the javascript file.

String Crime = “xxxxxx”; the “xxxxxx” is replaced with any of the comment items example “ATTEMPTED ROBBERY” to generate coordinates of that crime .

import java.util.\*;  
public class GetLocation  
{  
 public static void main(String[] args)  
 {  
 Scanner in = new Scanner(System.*in*);  
 //System.out.println("Select class : ");  
 //System.out.println("RAPE\nROBBERY\nATTEMPTED ROBBERY\nASSAULT WITH DEADLY WEAPON\nBURGLARY\nBURGLARY FROM VEHICLE\nTHEFT-GRAND ($950.01 & OVER)\nSHOPLIFTING-GRAND THEFT ($950.01 & OVER)\n");  
 //System.out.println("THEFT, PERSON\nTHEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)\nSHOPLIFTING - PETTY THEFT ($950 & UNDER)\nVEHICLE - STOLEN\nBATTERY - SIMPLE ASSAULT\nINDECENT EXPOSURE\nBATTERY WITH SEXUAL CONTACT\nEXTORTION\n");  
 //String Crime = in.nextLine();  
 String Crime = "KIDNAPPING";  
  
 ArrayList<String> CrimeDescription = new ArrayList<String>();  
 int[] CrimeCount = new int[25];  
 double count = 0.0;  
  
 String input ="";  
 while(true)  
 {  
 input = in.nextLine();  
 if(input.equals("stop"))  
 break;  
  
 StringTokenizer st = new StringTokenizer(input,",");  
 if(st.nextToken().equals(Crime))  
 { CrimeDescription.add("{lat: "+(st.nextToken())+", lng: "+(st.nextToken())+"},"); }  
 }  
  
 Random rand = new Random();  
 int N = 200;/\*  
 for(int i = 0; i < N; i++)  
 {   
 int n = rand.nextInt(CrimeDescription.size()) + 1;  
 System.out.println(CrimeDescription.get(n));   
 }/\*/  
 for(int i = 0; i < CrimeDescription.size(); i++)  
 {  
 //int n = rand.nextInt(CrimeDescription.size()) + 1;  
 if(i%10 == 0) System.*out*.println();  
 System.*out*.print(CrimeDescription.get(i)+"\t");  
 }  
 }  
}