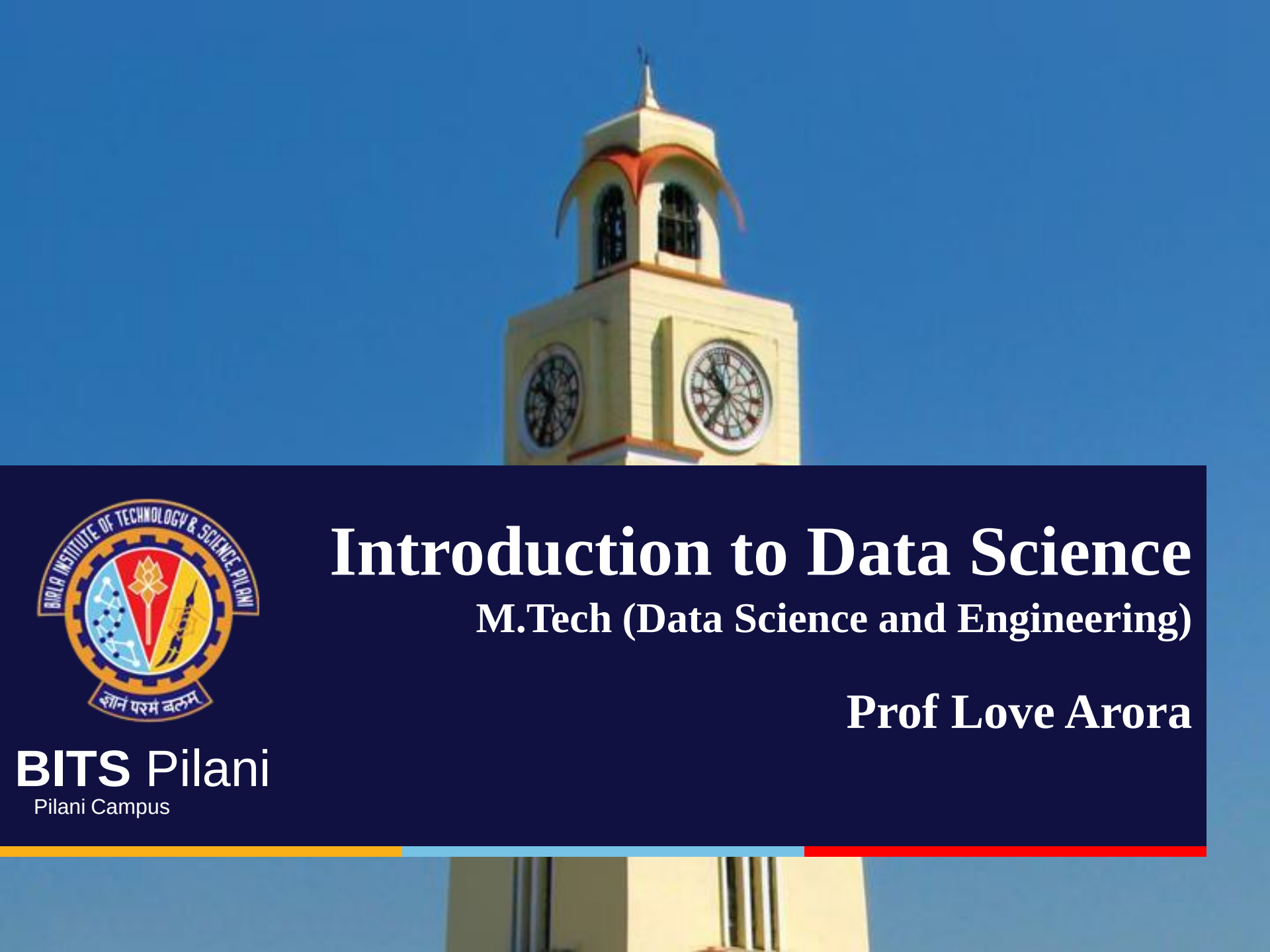


Introduction to Data Science

M.Tech (Data Science and Engineering)

Prof Love Arora

BITS Pilani

Pilani Campus

K-means Clustering

Overview

Clustering is an unsupervised learning technique. It is the task of grouping together a set of objects in a way that objects in the same cluster are more similar to each other than to objects in other clusters. Similarity is an amount that reflects the strength of relationship between two data objects. Clustering is mainly used for exploratory data mining. It is used in many fields such as machine learning, pattern recognition, image analysis, information retrieval, bio-informatics, data compression, and computer graphics.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Clustering: Types

Clustering can be broadly divided into two subgroups:

- Hard clustering: in hard clustering, each data object or point either belongs to a cluster completely or not.

- Soft clustering: in soft clustering, a data point can belong to more than one cluster with some probability or likelihood value. For example, you could identify some locations as the border points belonging to two or more regions.

Clustering Algorithms

Clustering algorithms can be categorized based on their cluster model, that is based on how they form clusters or groups.

- Connectivity-based clustering
- Centroid-based clustering
- Distribution-based clustering
- Density-based clustering

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Partitioning-based Clustering

Partitioning-based clustering: in this type of clustering, clusters are represented by a central vector or a centroid. This centroid might not necessarily be a member of the dataset. This is an iterative clustering algorithms in which the notion of similarity is derived by how close a data point is to the centroid of the cluster. k-means is a centroid based clustering, and will you see this topic more in detail later on in the tutorial.

Construct k partitions (k <= n) and then evaluate them by some criterion, e.g., minimizing the sum of square errors.

- Each group has at least one object, each object belongs to one group
- Iterative Relocation Technique
- Avoid Enumeration by storing the centroids
- Typical methods: k-means, k-medoids, CLARANS

Connectivity based Clustering

Connectivity-based clustering: the main idea behind this clustering is that data points that are closer in the data space are more related (similar) than to data points farther away. At different distances, different clusters will form and can be represented using a dendrogram, which gives away why they are also commonly called "hierarchical clustering". These methods do not produce a unique partitioning of the dataset, rather a hierarchy from which the user still needs to choose appropriate clusters by choosing the level where they want to cluster. They are also not very robust towards outliers, which might show up as additional clusters or even cause other clusters to merge.

Create a hierarchical decomposition of the set of data (or objects) using some criterion

- Agglomerative Vs Divisive
- Perform Analysis of linkages
- Integrate with iterative relocation
- Typical methods: Diana, Agnes, BIRCH

Density Based Clustering

Density-based methods search the data space for areas of varied density of data points. Clusters are defined as areas of higher density within the data space compared to other regions. Data points in the sparse areas are usually considered to be noise and/or border points. The drawback with these methods is that they expect some kind of density guide or parameters to detect cluster borders.

Distance based methods – Spherical Clusters

- Density – For each data point within a given cluster the neighbor hood should contain a minimum number of points
- DBSCAN, OPTICS

Distribution Based Clustering

Distribution-based clustering: this clustering is very closely related to statistics: distributional modeling. Clustering is based on the notion of how probable is it for a data point to belong to a certain distribution, such as the Gaussian distribution, for example. Data points in a cluster belong to the same distribution. These models have a strong theoretical foundation, however they often suffer from overfitting. Gaussian mixture models, using the expectation-maximization algorithm is a famous distribution based clustering method.

K-Mean Clustering Algorithm (Partitioning-based)

K-Means is probably the most well know clustering algorithm. It's easy to understand and implement in code!

K-Means has the advantage that it's pretty fast, as it is based on computing the distances between points and group centers; very few computations! It thus has a linear complexity $O(n)$.

K-Means has a couple of disadvantages.

- Firstly, you have to select how many groups/classes there are. This isn't always trivial and ideally with a clustering algorithm we'd want it to figure those out for us because the point of it is to gain some insight from the data
- K-means also starts with a random choice of cluster centers and therefore it may yield different clustering results on different runs of the algorithm

K-Mean Clustering Algorithm (Partitioning-based)

innovate achieve lead

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

K-Mean Clustering Algorithm (Partitioning-based)

- To begin, we first select a number of classes/groups to use and randomly initialize their respective center points. To figure out the number of classes to use, it's good to take a quick look at the data and try to identify any distinct groupings. The center points are vectors of the same length as each data point vector and are the "X's" in the graphic above.
- Each data point is classified by computing the distance between that point and each group center, and then classifying the point to be in the group whose center is closest to it.
- Based on these classified points, we recompute the group center by taking the mean of all the vectors in the group.
- Repeat these steps for a set number of iterations or until the group centers don't change much between iterations. You can also opt to randomly initialize the group centers a few times, and then select the run that looks like it provided the best results.

Other Clustering Algorithm

- **Linear clustering algorithm**
  - k-means clustering algorithm
  - Fuzzy c-means clustering algorithm
  - Hierarchical clustering algorithm
  - Gaussian(EM) clustering algorithm
  - Quality threshold clustering algorithm
- **Non-linear clustering algorithm**
  - MST based clustering algorithm
  - kernel k-means clustering algorithm
  - Density-based clustering algorithm

K-Means Calculations

Consider the below data set which has the values of the data points on a particular graph.

Table 1:

Documents (Data Points)	W1 (x-axis)	W2 (y-axis)
D1	2	0
D2	1	3
D3	3	5
D4	2	2
D5	4	6

We can randomly choose two initial points as the centroids and from there we can start calculating distance of each point.

For now we will consider that D2 and D4 are the centroids.

To start with we should calculate the distance with the help of Euclidean Distance which is

$$\sqrt{((x1-y1)^2 + (x2-y2)^2}$$

K-Means Calculations

innovate achieve lead

Iteration 1:

Step 1: We need to calculate the distance between the initial centroid points with other data points. Below I have shown the calculation of distance from initial centroids D2 and D4 from data point D1.

Distance between D1 and D2	Distance between D1 and D4
$\sqrt{(2-1)^2+(0-3)^2}$	$\sqrt{(2-2)^2+(0-2)^2}$
$= \sqrt{(1)^2+(3)^2}$	$= \sqrt{(0)^2+(-2)^2}$
$= \sqrt{1+9}$	$= \sqrt{0+4}$
$= \sqrt{10} = 3.17$	$= \sqrt{4} = 2$

After calculating the distance of all data points, we get the values as below.

Table 2:

Documents (Data Points)	Distance between D2 and other data points	Distance between D4 and other data points
D1	3.17	2.0
D3	2.83	3.17
D5	4.25	4.48

K-Means Calculations

innovate achieve lead

Step 2: Next, we need to group the data points which are closer to centriods. Observe the above table, we can notice that D1 is closer to D4 as the distance is less. Hence we can say that D1 belongs to D4 Similarly, D3 and D5 belongs to D2. After grouping, we need to calculate the mean of grouped values from Table 1.

Cluster 1: (D1, D4) Cluster 2: (D2, D3, D5)

Step 3: Now, we calculate the mean values of the clusters created and the new centriod values will these mean values and centroid is moved along the graph.

Clusters	Mean value of data points along x -axis	Distance between D4 and other data points
D1, D4	2.0	1.0
D2, D3, D5	2.67	4.67

From the above table, we can say the new centroid for cluster 1 is (2.0, 1.0) and for cluster 2 is (2.67, 4.67)

K-Means Calculations

innovate achieve lead

Iteration 2:

Step 4: Again the values of euclidean distance is calculated from the new centriods. Below is the table of distance between data points and new centroids.

Documents (Data Points)	Distance between centroid of cluster 1 and data points	Distance between centroid of cluster 2 and data points
D1	1.0	4.72
D2	2.24	2.37
D3	4.13	0.47
D4	1	2.76
D5	5.39	1.89

We can notice now that clusters have changed the data points. Now the cluster 1 has D1, D2 and D4 data objects. Similarly, cluster 2 has D3 and D5

Step 5: Calculate the mean values of new clustered groups from Table 1 which we followed in step 3. The below table will show the mean values

Clusters	Mean value of data points along x -axis	Distance between D4 and other data points
D1, D2, D4	1.67	1.67
D3, D5	3.5	5.5

Now we have the new centroid value as following:

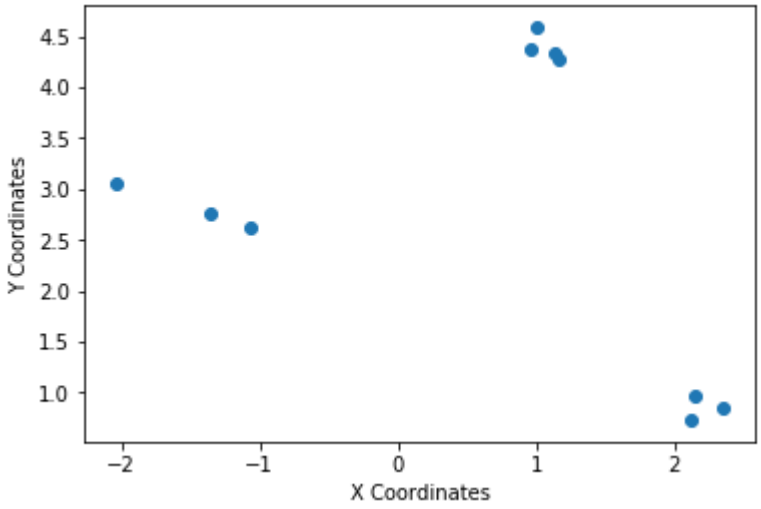
cluster 1 (D1, D2, D4) - (1.67, 1.67) and cluster 2 (D3, D5) - (3.5, 5.5)

This process has to be repeated until we find a constant value for centroids and the latest cluster will be considered as the final cluster solution.

K-Means Code Snippet


```
# Imports
from sklearn.datasets.samples_generator import make_blobs

# Generate 2D data points
X, _ = make_blobs(n_samples=10, centers=3, n_features=2,
                  cluster_std=0.2,  random_state=0)

# Convert the data points into a pandas DataFrame
import pandas as pd

# Generate indicators for the data points
obj_names = []
for i in range(1, 11):
    obj = "Object " + str(i)
    obj_names.append(obj)

# Create a pandas DataFrame with the names and (x, y) coordinates
data = pd.DataFrame({
    'Object': obj_names,
    'X_value': X[:, 0],
    'Y_value': X[:, -1]
})

# Preview the data
print(data.head())
```

	Object	X_value	Y_value
0	Object 1	1.005079	4.594642
1	Object 2	1.128478	4.328122
2	Object 3	2.117881	0.726845
3	Object 4	0.955626	4.385907
4	Object 5	-1.354017	2.769449



K-Means Code Snippet

innovate achieve lead

```
# Initialize the centroids
c1 = (-1, 4)
c2 = (-0.2, 1.5)
c3 = (2, 2.5)

# A helper function to calculate the Euclidean diatance between the data
# points and the centroids

def calculate_distance(centroid, X, Y):
    distances = []

    # Unpack the x and y coordinates of the centroid
    c_x, c_y = centroid

    # Iterate over the data points and calculate the distance using the          # given formula
    for x, y in list(zip(X, Y)):
        root_diff_x = (x - c_x) ** 2
        root_diff_y = (y - c_y) ** 2
        distance = np.sqrt(root_diff_x + root_diff_y)
        distances.append(distance)

    return distances
```

K-Means Code Snippet

innovate achieve lead

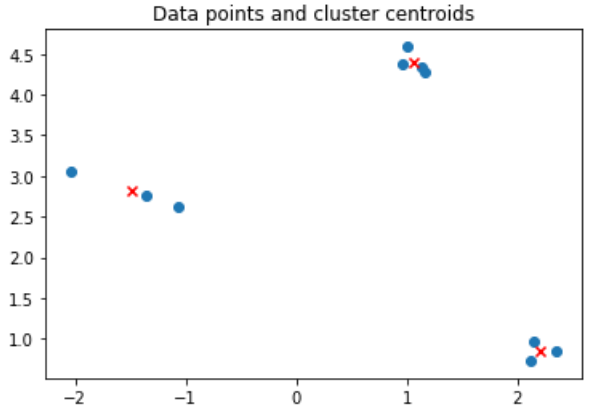
```
# Calculate the distance and assign them to the DataFrame accordingly
data['C1_Distance'] = calculate_distance(c1, data.X_value, data.Y_value)
data['C2_Distance'] = calculate_distance(c2, data.X_value, data.Y_value)
data['C3_Distance'] = calculate_distance(c3, data.X_value, data.Y_value)

# Preview the data
print(data.head())

# Get the minimum distance centroids
   data['Cluster'] = data[['C1_Distance', 'C2_Distance', 'C3_Distance']].apply(np.argmin, axis =1)

   # Map the centroids accordingly and rename them
   data['Cluster'] = data['Cluster'].map({'C1_Distance': 'C1', 'C2_Distance': 'C2', 'C3_Distance': 'C3'})

   # Get a preview of the data
   print(data.head(10))

# Calculate the coordinates of the new centroid from cluster 1
x_new_centroid1 = data[data['Cluster']=='C1']['X_value'].mean()
y_new_centroid1 = data[data['Cluster']=='C1']['Y_value'].mean()
# Calculate the coordinates of the new centroid from cluster 2
x_new_centroid2 = data[data['Cluster']=='C3']['X_value'].mean()
y_new_centroid2 = data[data['Cluster']=='C3']['Y_value'].mean()
# Print the coordinates of the new centroids
print('Centroid 1 ({}, {})'.format(x_new_centroid1, y_new_centroid1))
print('Centroid 2 ({}, {})'.format(x_new_centroid2, y_new_centroid2))
```

K-Means- scikit

```
# Using scikit-learn to perform K-Means clustering
from sklearn.cluster import KMeans

# Specify the number of clusters (3) and fit the data X
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)

# Get the cluster centroids
print(kmeans.cluster_centers_)

# Get the cluster labels
print(kmeans.labels_)

# Plotting the cluster centers and the data points on a 2D plane
plt.scatter(X[:, 0], X[:, -1])

plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], c='red', marker='x')

plt.title('Data points and cluster centroids')
plt.show()
```

```
array([[-1.48815728,  2.81502145],
       [ 2.20533486,  0.84929168],
       [ 1.06386762,  4.39554682]])
```

```
array([2, 2, 1, 2, 0, 0, 0, 1, 1, 2])
```



Case Study : Background

Uber dataset, which contains data generated by Uber for the city on New York. Uber Technologies Inc. is a peer-to-peer ride sharing platform. Don't worry if you don't know too much about Uber, all you need to know is that the Uber platform connects you with (cab)drivers who can drive you to your destiny. The data is freely available on Kaggle. The dataset contains raw data on Uber pickups with information such as the date, time of the trip along with the longitude-latitude information.

New York city has five boroughs: Brooklyn, Queens, Manhattan, Bronx, and Staten Island. At the end of this mini-project, you will apply k-means clustering on the dataset to explore the dataset better and identify the different boroughs within New York. All along, you will also learn the various steps that you should take when working on a data science project

Case Study : Problem Statement

There is a lot of information stored in the traffic flow of any city. This data when mined over location can provide information about the major attractions of the city, it can help us understand the various zones of the city such as residential areas, office/school zones, highways, etc. This can help governments and other institutes plan the city better and enforce suitable rules and regulations accordingly. For example, a different speed limit in school and residential zone than compared to highway zones.

The data when monitored over time can help us identify rush hours, holiday season, impact of weather, etc. This knowledge can be applied for better planning and traffic management. This can at a large, impact the efficiency of the city and can also help avoid disasters, or at least faster redirection of traffic flow after accidents.

Case Study : Result