Anaomaliy detection using K Means Clustering

```
#Part A: Detection on a small dataset

#Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
```

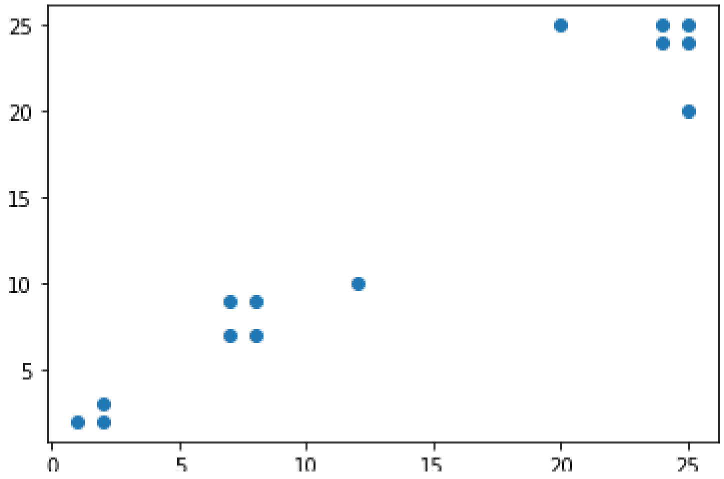
```
#Generate the daatset (a numpy array)
data = np.array([[1,2],[2,2],[2,3],[8,7],[8,9],[7,9],[7,7],[12,10],[25,24],[24,24],[24,25]
```

```
data
```

```
     array([[ 1,  2],
            [ 2,  2],
            [ 2,  3],
            [ 8,  7],
            [ 8,  9],
            [ 7,  9],
            [ 7,  7],
            [12, 10],
            [25, 24],
            [24, 24],
            [24, 25],
            [25, 25],
            [25, 20],
            [20, 25]])
```

```
# Plot the data
plt.scatter(data[:,0],data[:,1])
```

```
     <matplotlib.collections.PathCollection at 0x7f208e36e290>
```



```
#k means model with k=3
km = KMeans(n_clusters=3)
```

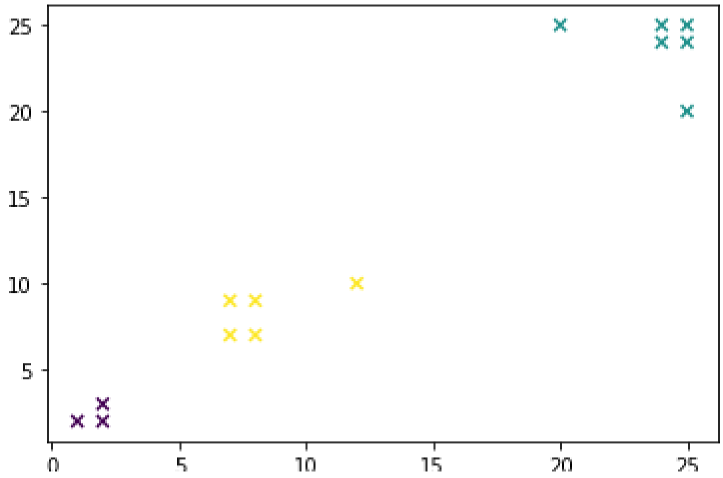
```
clusters = km.fit_predict(data)
```

```
clusters
```

```
array([0, 0, 0, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1], dtype=int32)
```

```
#plotting the clusters
plt.scatter(data[:,0],data[:,1], c=clusters, marker = "x")
```

```
<matplotlib.collections.PathCollection at 0x7f208de5f090>
```



```
#obtain the center of the clusters
centroids = km.cluster_centers_
```

```
centroids
```

```
array([[ 1.66666667,  2.33333333],
       [23.83333333, 23.83333333],
       [ 8.4       ,  8.4       ]])
```

```
#initialize an array which will be used to reach the index
points = np.empty((0,len(data[0])),float)
points
```

```
array([], shape=(0, 2), dtype=float64)
```

```
#initialise an array to append the distances of the points from the centroids
distances = np.empty((0,len(data[0])),float)
distances
```

```
array([], shape=(0, 2), dtype=float64)
```

```
for i, center_elem in enumerate(centroids):
  distances = np.append(distances,cdist([center_elem], data[clusters==i], 'euclidean'))
  points = np.append(points, data[clusters==i], axis=0)
```

```
distances
```

```
array([0.74535599, 0.47140452, 0.74535599, 1.1785113 , 0.23570226,
       1.1785113 , 1.64991582, 4.00693843, 4.00693843, 1.45602198,
       0.72111026, 1.52315462, 1.97989899, 3.93954312])
```

```
points
```

```
array([[ 1.,  2.],
       [ 2.,  2.],
       [ 2.,  3.],
       [25., 24.],
       [24., 24.],
       [24., 25.],
       [25., 25.],
       [25., 20.],
       [20., 25.],
       [ 8.,  7.],
       [ 8.,  9.],
       [ 7.,  9.],
       [ 7.,  7.],
       [12., 10.]])
```

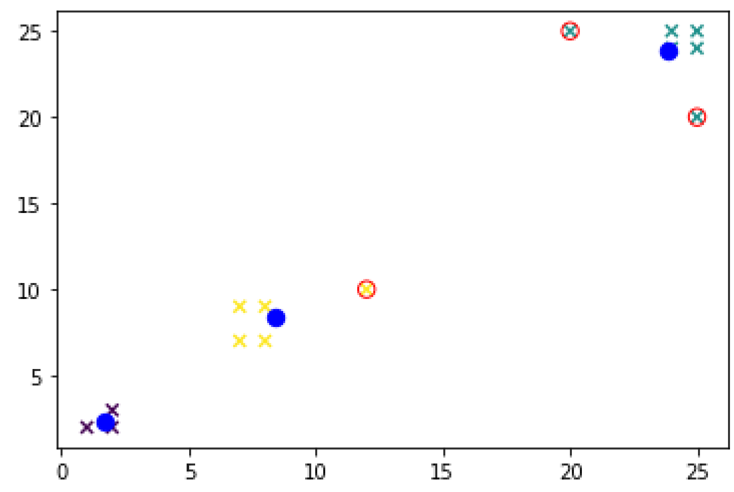
```
threshold = 80
#get those data points whose distances from teir cluster centers which is greater than the
outliers = points[np.where(distances>np.percentile(distances,threshold))]
```

```
outliers
```

```
array([[25., 20.],
       [20., 25.],
       [12., 10.]])
```

```
plt.scatter(data[:,0],data[:,1], c=clusters, marker = "x")
plt.scatter(outliers[:,0],outliers[:,1], marker = "o", facecolor='None', edgecolors="r", s
plt.scatter(centroids[:,0],centroids[:,1], marker = "o", facecolor='b', s=70)
```

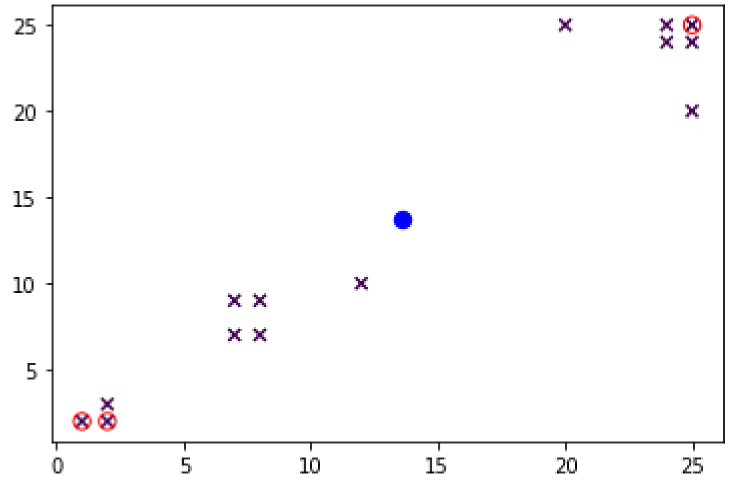
```
<matplotlib.collections.PathCollection at 0x7f208ddedb10>
```



```
#obtain the center of the clusters
```

```
#k means model with k=1
km = KMeans(n_clusters=1)
clusters = km.fit_predict(data)
clusters
centroids = km.cluster_centers_
centroids
#initialize an array which will be used to reach the index
points = np.empty((0,len(data[0])),float)
points
#initialise an array to append the distances of the points from the centroids
distances = np.empty((0,len(data[0])),float)
distances
for i, center_elem in enumerate(centroids):
  distances = np.append(distances,cdist([center_elem], data[clusters==i], 'euclidean'))
  points = np.append(points, data[clusters==i], axis=0)

threshold = 80
#get those data points whose distances from teir cluster centers which is greater than the
outliers = points[np.where(distances>np.percentile(distances,threshold))]

plt.scatter(data[:,0],data[:,1], c=clusters, marker = "x")
plt.scatter(outliers[:,0],outliers[:,1], marker = "o", facecolor='None', edgecolors="r", s
plt.scatter(centroids[:,0],centroids[:,1], marker = "o", facecolor='b', s=70)
```

```
<matplotlib.collections.PathCollection at 0x7f208dd68610>
```



Part B

Anamoly Detection on a random generated regression dataset

```
from numpy import sqrt, array, random, argsort
```

```
random.seed(121)
```

```
#function to prepare the dataset
def makeData(N):
  x=[]
  for i in range(N):
    a=i/1000 +random.uniform(-3,2)
    r=random.uniform(-5,10)
    if (r>=9.9):
      r=r+10
    elif(r<(-4.8)):
      r=r+(-10)
    x.append([a+r])
  return array(x)
```

```
x=makeData(500)
```

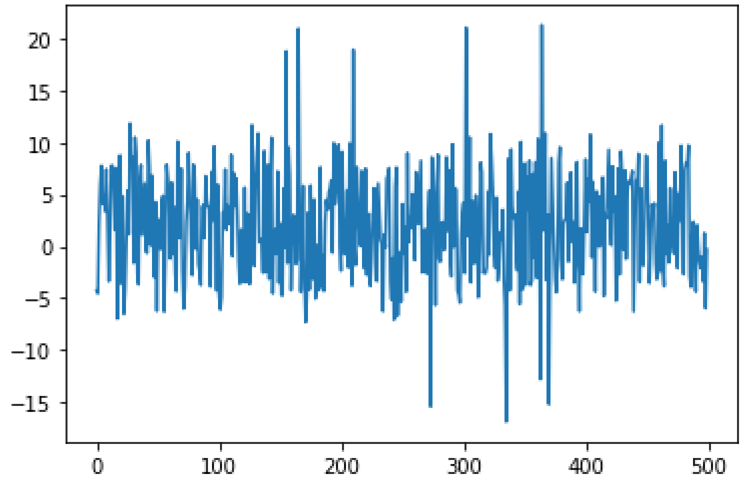
```
x.shape
```

```
(500, 1)
```

```
x_ax = range(500)
```

```
plt.plot(x_ax,x)
```

```
[<matplotlib.lines.Line2D at 0x7f208de7cf50>]
```



```
#standardize the data
from sklearn.preprocessing import scale
x= scale(x)
```

```
#fit the kmeans model
kmeans= KMeans(n_clusters =1).fit(x)
```

```
#centroid of the fitted model
center = kmeans.cluster_centers_
print(center)
```

```
     [[-1.95399252e-17]]
```

```
#distance of each data point from the central value
distance = sqrt((x-center)**2)


#sort the distance
order_index = argsort(distance, axis=0)


#extract those data points with the largest distance
indexes = order_index[-10:]
values = x[indexes]
```

```
indexes
```

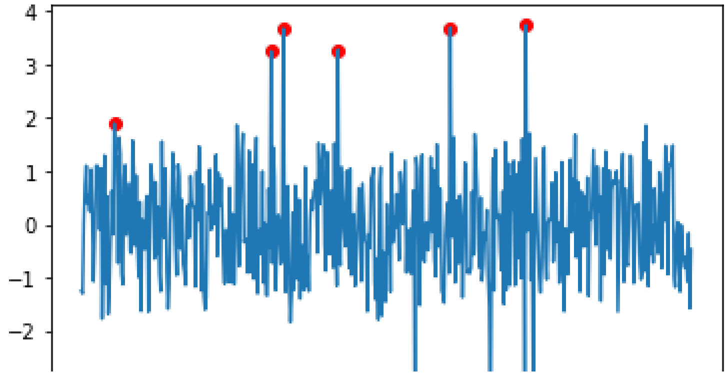
```
     array([[ 27],
            [363],
            [155],
            [210],
            [370],
            [273],
            [165],
            [302],
            [335],
            [364]])
```

```
values
```

```
     array([[[ 1.90165599]],

            [[-2.90588858]],

            [[ 3.24957092]],

            [[ 3.27848142]],

            [[-3.37427853]],

            [[-3.41691761]],

            [[ 3.6755354 ]],

            [[ 3.69435162]],

            [[-3.70515047]],

            [[ 3.74189719]]])
```

```
plt.plot(x_ax,x)
plt.scatter(indexes, values, color="r")
```

```
<matplotlib.collections.PathCollection at 0x7f2089501310>
```



Part C

Anamoly detection on Boston data set by applying k means

```
from sklearn.datasets import load_boston
```

```
boston = load_boston()
y=boston.target
y=y.reshape(y.shape[0],1)
y=scale(y)
```

```
#fit the kmeans model
kmeans= KMeans(n_clusters =1).fit(x)
```

```
#centroid of the fitted model
center = kmeans.cluster_centers_
print(center)
```

```
    [[-1.95399252e-17]]
```

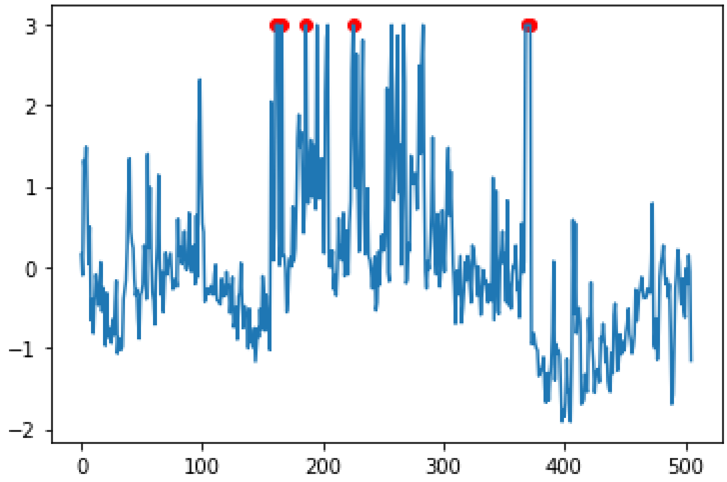
```
#distance of each data point from the central value
distance = sqrt((y-center)**2)
```

```
#sort the distance
order_index = argsort(distance, axis=0)
```

```
#extract those data points with the largest distance
indexes = order_index[-10:]
values = y[indexes]
```

```
x_ax = range(y.shape[0])
plt.plot(x_ax,y)
plt.scatter(indexes, values, color="r")
```

```
<matplotlib.collections.PathCollection at 0x7f20887deed0>
```


3
2
1
0
−1
−2
0
100
200
300
400
500