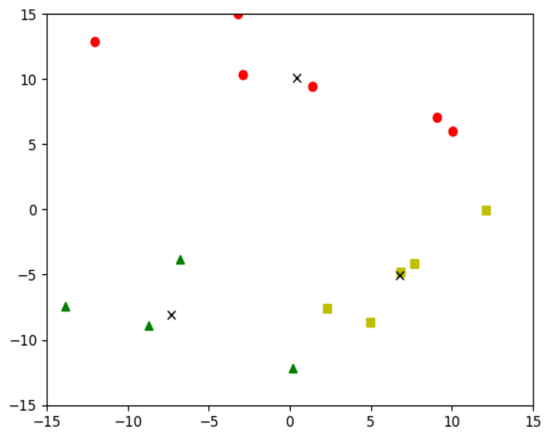
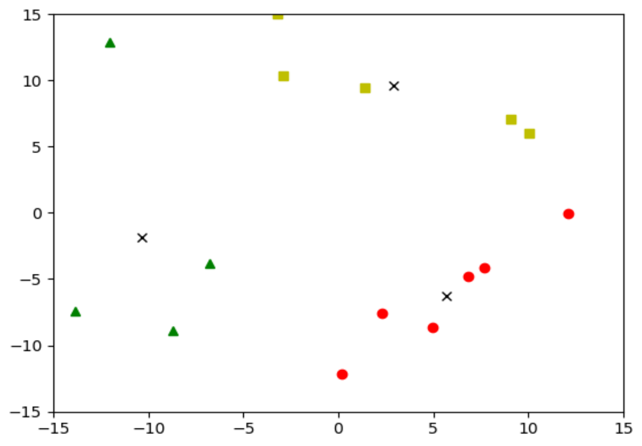
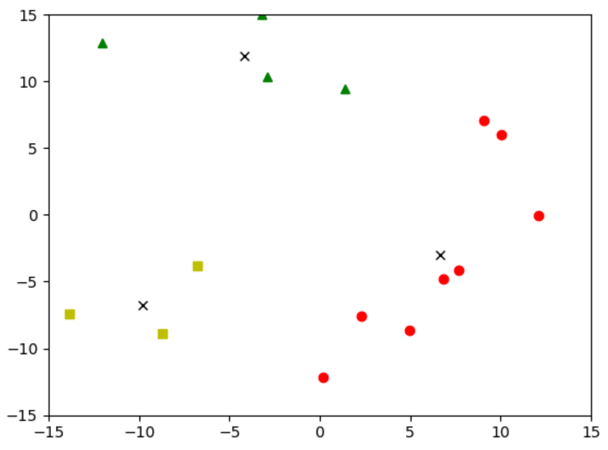
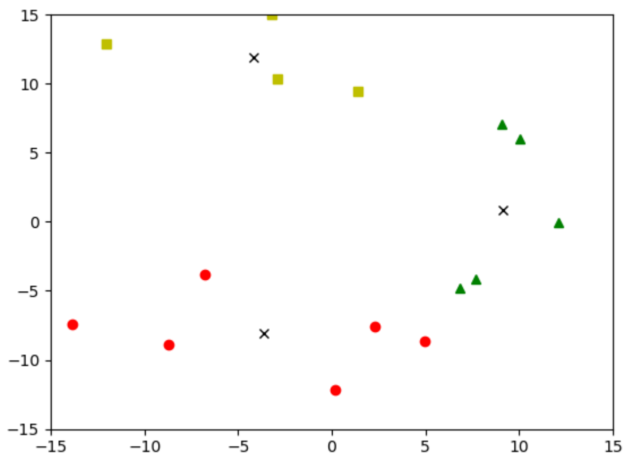
**Task I:**Kmeans clustering is a type of unsupervised learning used when we need to classify unlabelled data. The algorithms task is to split the data into k no. of cluster depending on the centroid/means of selected points. Running K means on random data to find cluster centers.

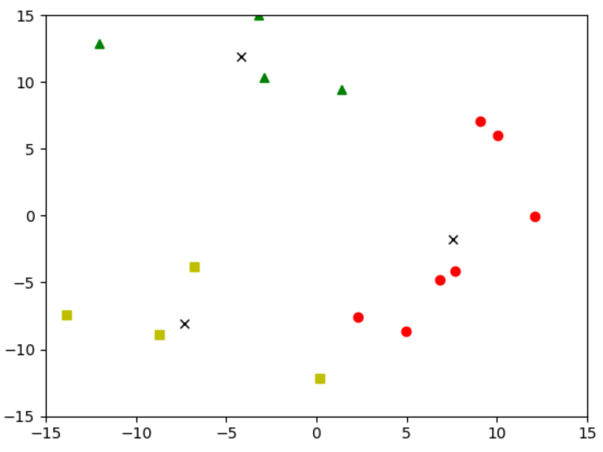
Random set evaluation carried out on:
```
[[   6.84615961 ,-4.79164637],
 [-13.86192175 , -7.45096698],
 [  1.40224514 ,  9.41272206],
 [  7.66997853 , -4.12972197],
 [ -8.73778734 , -8.92961497],
 [ -3.20476509 , 14.9798459 ],
 [  0.19172171 ,-12.15434282],
 [  9.08760577  , 7.08445964],
 [  4.9688782   ,-8.64113377],
 [ -2.87381972  ,10.37677759],
 [-12.05726807  ,12.89207442],
 [ 12.07092633  ,-0.06560177],
 [ 10.04890739  , 6.00827876],
 [ -6.7764187   , -3.83422637],
 [  2.2772154   , -7.61583694]])
```

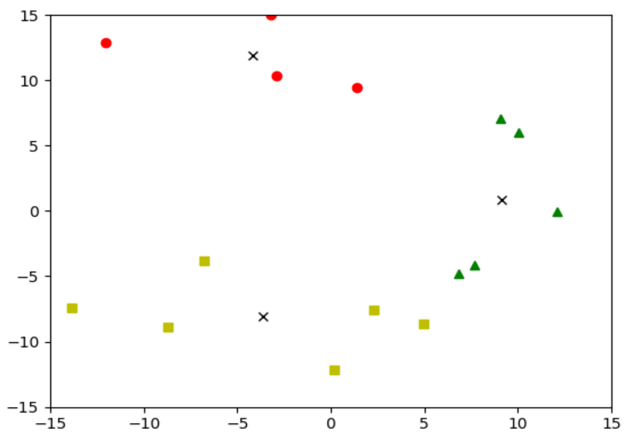












The top cluster mean is classified correct almost in all the tested iterations. However there is some slight overlap with the right side cluster mean. There are only two variations of the cluster formed in this method. The topleft image and the middle left image depict the different cluster means obtained. This leads to the resultant 2 points laying in either the top or the right class.

On the other hand the bottom cluster and the right side cluster have significant overlap of points among them. This leads to different variations in almost all the tested iterations. Overall the middle left iteration seems to have the best cluster results.

**TaskII:**

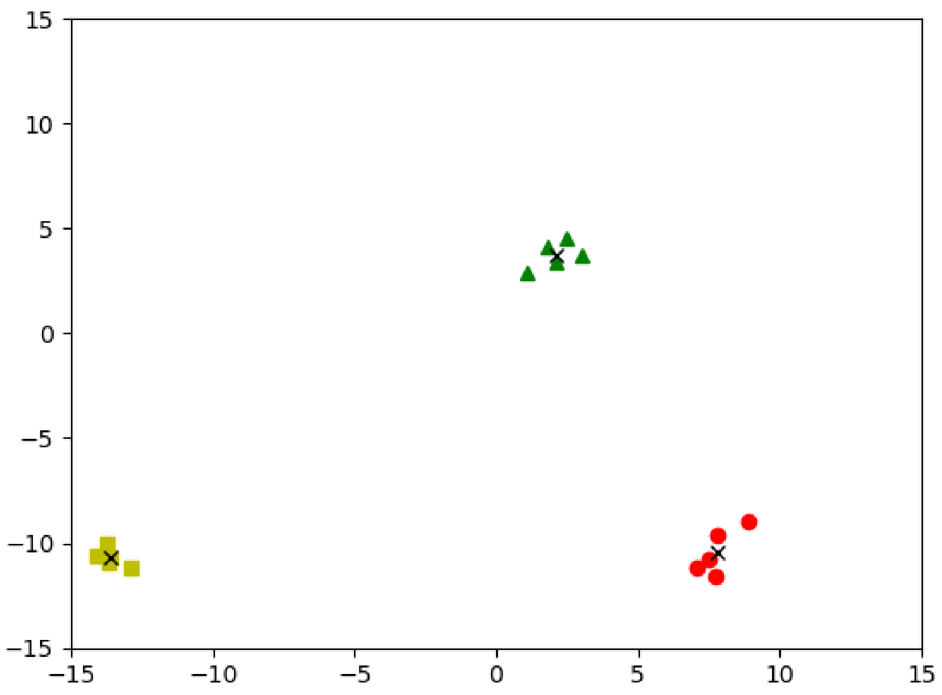
Data used for tight clusters (x,y) pairs:

```
[[  2.12 ,3.4],
[3 , 3.7],
[  2.5,  4.5],
[  1.1 , 2.9],
[ 1.8 , 4.1],
[ -13.6 , -10.7 ],
[  -13.7 ,-10],
[  -13.65  ,-10.9],
[  -14.1  ,-10.6],
[ -12.9  ,-11.21],
[7.5  ,-10.8],
[ 7.77  ,-11.6],
[ 7.80 , -9.6],
[ 7.1  , -11.2],
[  8.90  , -8.98]]
```

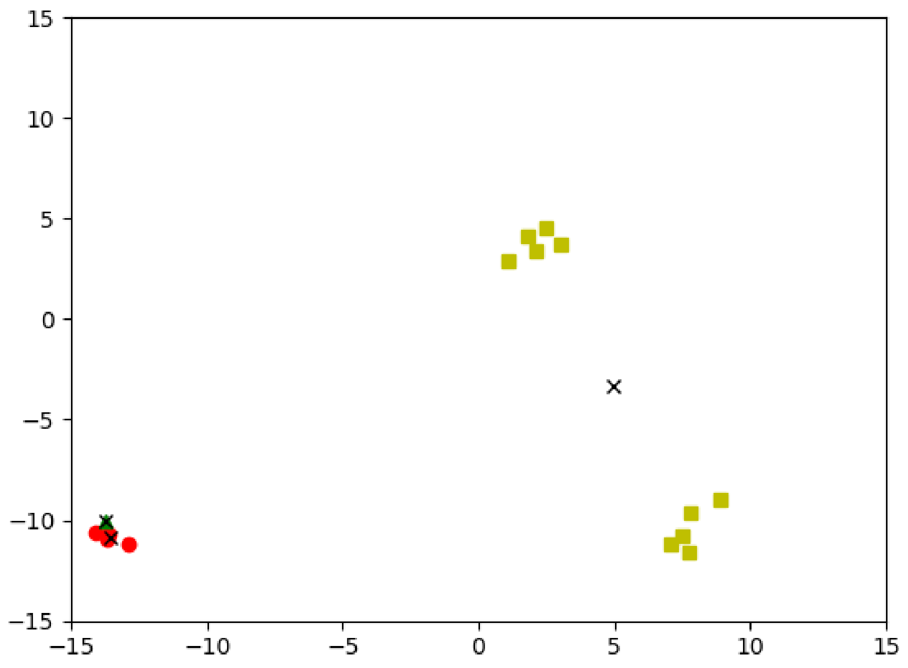
We get clusters as(each row is a set of three x,y pairs of cluster centers):

```
iteration 0 = [2.104 3.72 ] [  7.814 -10.436] [-13.59  -10.682]
iteration 1 = [  7.814 -10.436] [-13.59  -10.682] [2.104 3.72 ]
iteration 2 = [  7.814 -10.436] [-13.59  -10.682] [2.104 3.72 ]
iteration 3 = [2.104 3.72 ] [  7.814 -10.436] [-13.59  -10.682]
iteration 4 = [-13.59  -10.682] [  7.814 -10.436] [2.104 3.72 ]
iteration 5 = [2.104 3.72 ] [-13.59  -10.682] [  7.814 -10.436]
iteration 6 = [2.104 3.72 ] [  7.814 -10.436] [-13.59  -10.682]
iteration 7 = [ 4.959 -3.358] [-13.7 -10. ] [-13.5625 -10.8525]
iteration 8 = [-13.9 -10.3] [ 4.959 -3.358] [-13.38333333 -10.93666667]
iteration 9 = [ 4.959 -3.358] [-13.9 -10.3] [-13.38333333 -10.93666667]
iteration 10 = [-13.59  -10.682] [  7.814 -10.436] [2.104 3.72 ]
iteration 11 = [2.104 3.72 ] [-13.59  -10.682] [  7.814 -10.436]
iteration 12 = [  7.814 -10.436] [-13.59  -10.682] [2.104 3.72 ]
iteration 13 = [2.104 3.72 ] [  7.814 -10.436] [-13.59  -10.682]
iteration 14 = [-13.5625 -10.8525] [-13.7 -10. ] [ 4.959 -3.358]
iteration 15 = [2.104 3.72 ] [-13.59  -10.682] [  7.814 -10.436]
iteration 16 = [2.104 3.72 ] [-13.59  -10.682] [  7.814 -10.436]
iteration 17 = [-13.7625 -10.55  ] [-12.9  -11.21] [ 4.959 -3.358]
iteration 18 = [2.104 3.72 ] [  7.814 -10.436] [-13.59  -10.682]
iteration 19 = [-13.38333333 -10.93666667] [ 4.959 -3.358] [-13.9 -10.3]
```

On an average around 17 out of the 20 iterations result in clusters in the following fashion:



However around 2-4 iterations result in the means as :



This result is obviously not intended by the algorithm. The two bottom left clusters around -14,-11 are too close to each other. The third cluster centre is significantly far from any of its members. This is one of the major limitations of kmeans.

**TaskIII:**
The problem to find cluster centers is a NP-hard problem. Due to this issue Kmeans algorithm generally relies on heuristic calculations to arrive at the right means. In the most common kmeans technique we use iterative technique. We chose initial centres randomly (usually) and then we update the centres as we do more iterations on the data. This update takes place based on the Euclidean distance between the different points. This selection however only proceeds towards the local minimum. As it is a heuristic we can’t surely decide the centres in one iteration. The centres may depend on the initial selected clusters.

The most simple method to fix this issue is to run it multiple times to determine the true results of  kmeans for validation. This is possible because kmeans algorithm is fast even on big datasets. We can inspect the cluster centers that are correct by inspection if the clusters are tight and calculate the average centers of the respective cases.. However if the clusters are not tightly bound then we have to rely on different algorithms for accurate clusters.

There are many variations and alternatives for kmeans. For example: kmediods algorithm. This technique chooses the data points as the center. It measures a sum of pairwise dissimilarities instead of the sum squared Euclidean Distance. It is more robust to noise and outliers as compared to kmeans.

We can also vary the kmeans algorithm. One such variation: Lloyd’s algorithm. In this technique we split the dataspace into uniform convex areas. A technique called Voronoi diagrams is used instead of simply testing the Euclidean distance