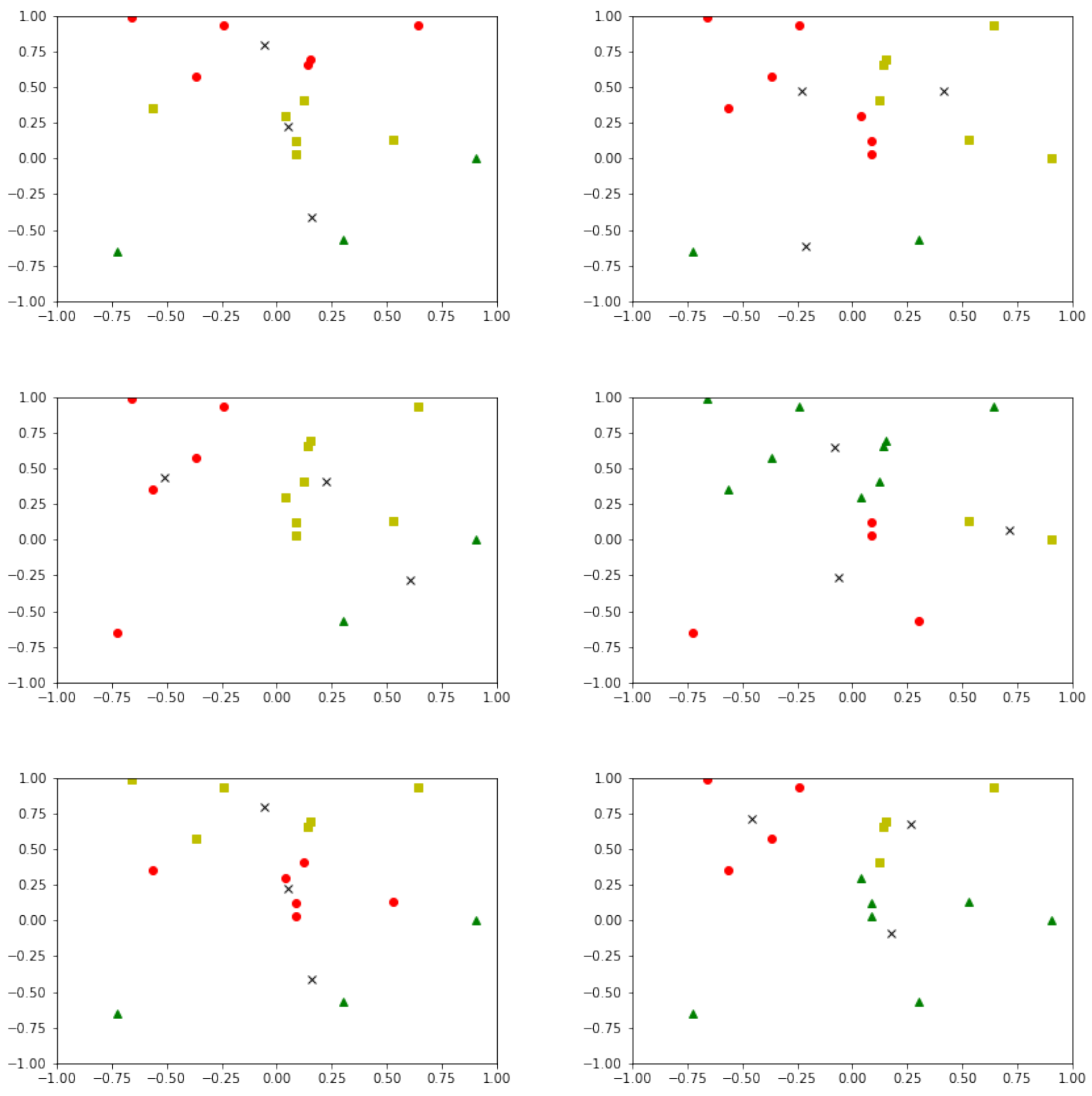
1 Exercise 1

This is the answer to question 1.

1. k-means is an unsupervised clustering algorithm used when an analyst wishes to identify clusters in data which may not be readily labelled. K-means is commonly used in the realms of digital marketing, CRM, and e-commerce where analysts may wish to identify customer segments within their data (e.g. NPS responses, transactional data, demographic information) which can be used for personalised marketing methods.; this data may lack explicit labels and by using methods such as k-means similar customer segments can be identified with proper data processing and preparation. k clusters are generated by forming n-dimensional balls (called clusters) of a given metric (commonly the minkowski metric) at a center such that the distance between clusters is maximised while the centroid of the cluster is minimised. A key limitation of k-means is that due to the generation of an n-dimensional ball around its center, non-convex clusters can be challenging to identify using naive k-means alone, while because the identification of a center is NP-hard, the identification of the optimal k to use (often informed via elbow plots) can be challenging and subsequent iterations on the same dataset can generate different clusters.

2. Using the provided k-means code, I generate the following plots:



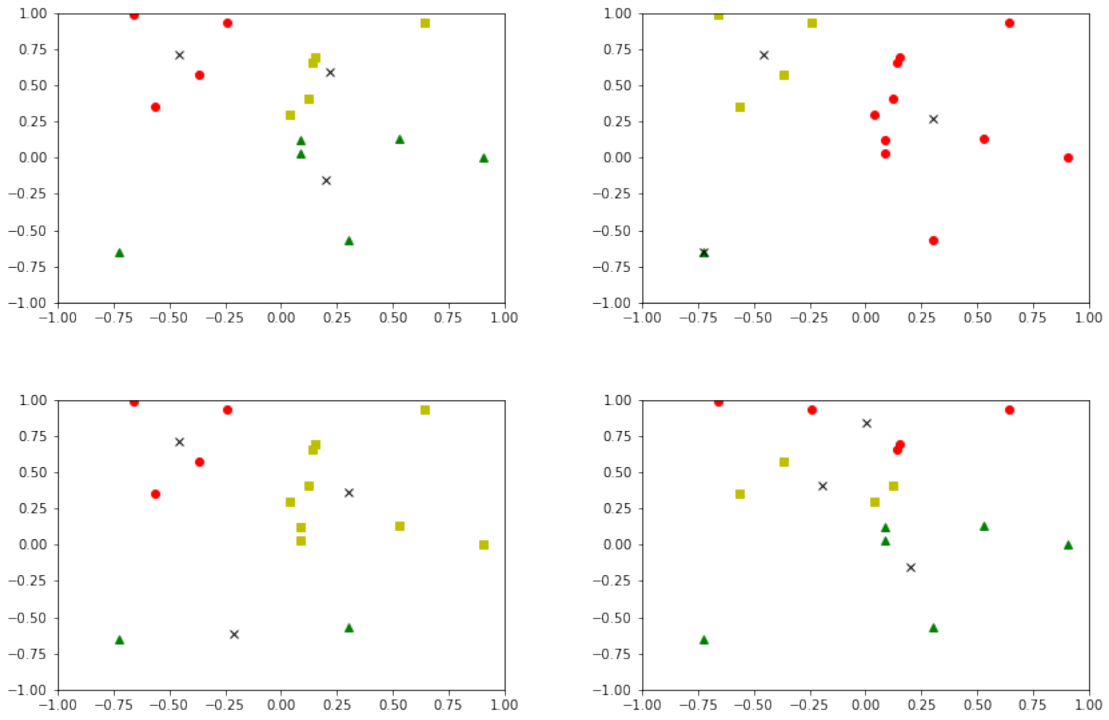


Figure 1: Question 1 Clusters

3. We see that although the clusters are different, there are similarities present in regards to where the centers are typically located and which points are assigned to which cluster. Very broadly, potential sets of clusters which have been identified are

   - A cluster involving a center involving the 'top' data in the plot, a cluster involving the 'middle' data in the plot, and a cluster involving the 'bottom' data in the plot.
   - A cluster involving data in the upper left, a cluster involving data in the upper right, and a cluster involving data in the bototm portion.

   While the points assigned to the clusters varies depending on the center which is assigned, broadly speaking each of the runs generate clusters which can be broadly classified into the above two types. We see the data is relatively mixed so 'clean' clusters are not so easily identified even by 'human' inspection of the data, and with the exception of the run which generated a center at point $(-0.72365683, -0.65275731)$, most candidates appear to identify feasible separate clusters. In reality, identifying which set was more feasible, or whether a different k other than 3 should be chosen, would depend on the context of what the x and y data represents and potentially information from other data associated with each point.

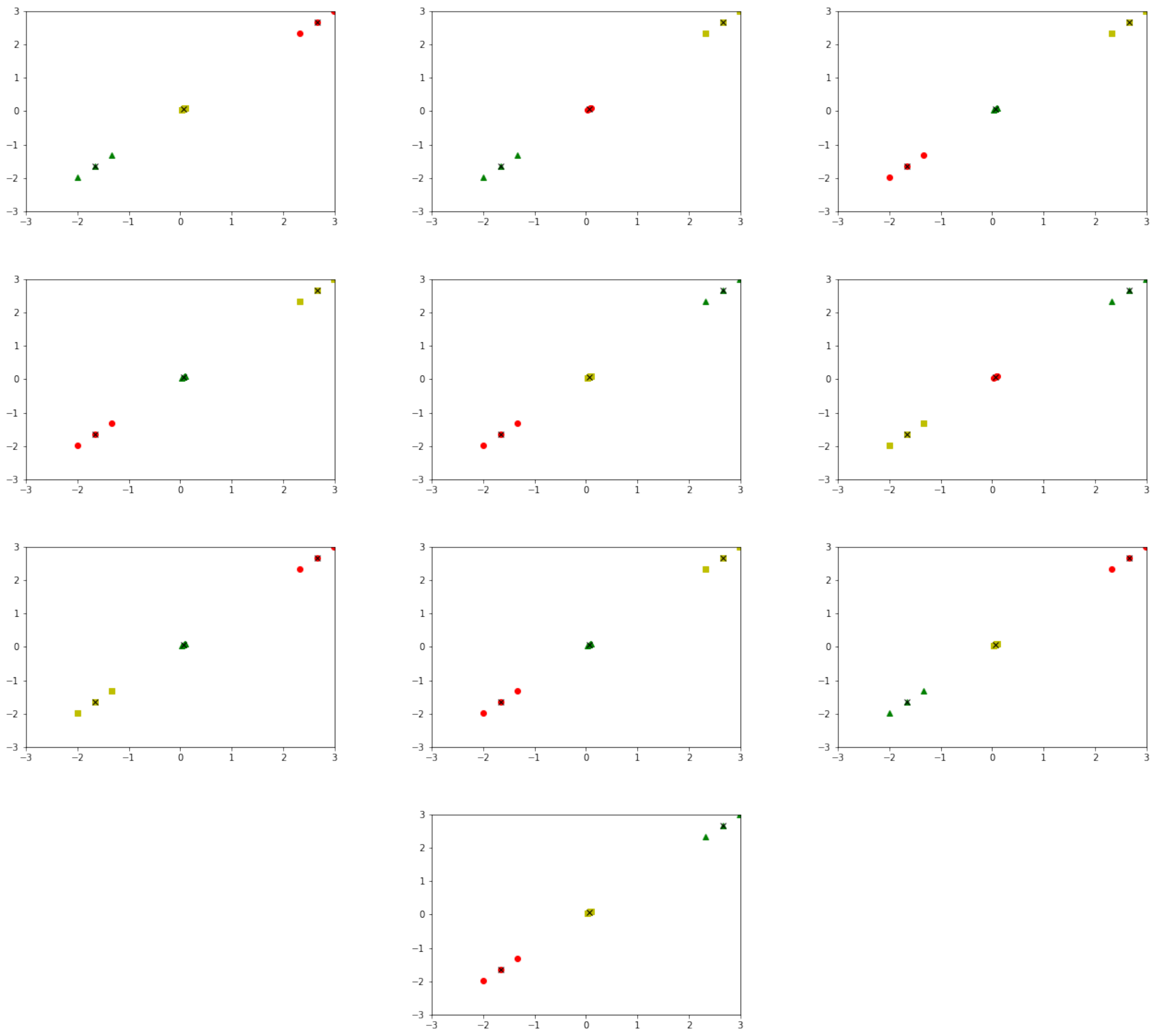
2 Exercise 2

This is the answer to question 2.

1. I chose the following data points 10 these exhibit a broadly linear relationship with clear clusters around the mid-point of each datapoints which share the same floor (0,2,-2):

```
[0.033,  0.033 ]
,[0.066,  0.066 ]
,[ 0.099,  0.099]
,[2.33, 2.33]
,[2.66,  2.66]
,[2.99 ,  2.99]
,[ -1.33,  -1.33]
,[ -1.99,  -1.99]
,[-1.66,  -1.66]
,[ 2.45, 2.45]]
```

2. Running the code on these datapoints I generate the following graphs:



3. Unexpectedly, although nine of the runs correctly identified the clusters, the very first run misidentified the clusters by putting two clusters into set of points with floor x_i of 2. By replacing the point $(2.45, 2.45)$ with the point $(0.077, 0.077)$ which has a low distance from the center of the middle cluster, I generate a dataset which when ran ten times is classified

correctly in each run. This change eliminates the potential for two centers in the upper-right cluster (2.33 and 2.45, and 2.66 and 2.99) and creates three natural and distinct clusters between each of the points which are well-separated with low variance within each cluster resulting in a correct classification.

3 Exercise 3

This is the answer to question 3.

1. The key challenge which is faced by k-means is that the centers are randomly placed depending on the seed and subsequently moved to minimise the distances, however this results in convergence to a local maximum rather than a global maximum (and due to the NP-hard nature of this, it is not 'easily' resolved.

2. One proposed enhancement to k-means is the k-means++ algorithm proposed by David Arthur and Sergei Vassilvitskii [1]. Broadly speaking, while k-means chooses the centers randomly, k-means++ generates the first center randomly and then, based on this first center, attempts to place the other centers. This paper uses monte-carlo simulation to show that this method can prove to be both faster and more consistent in some simulations than the traditional k-means algorithm. Although there is no guarantee for this to be the case, this is one such proposal which is commonly used to help eliminate some of the deviation which is provided by traditional k-means. This method is more algorithmic than the alternative solution of applying a static random seed in an attempt to 'work around' the 'randomness' of the starting point of k-means which, athough will result in consistent clusters, does not mean the produced clusters are 'better'. The usage of k-means++ is in fact what sklearn uses as its default algorithm in an attempt to both speed up convergence and produce more consistent results as per the KMeans documentation.

3. A key limitation of the KMeans++ algorithm is that because of the sequential nature of the algorithm in the initialisation process, the method can face challenges when dealing with massive datasets [2] and it is an on-going area of research to continuously improve the performance of kmeans++ in the domain of big data due to the methods' popularity and widespread usage.

References

[1] David Arthur and Sergei Vassilvitskii. *K-Means++: The Advantages of Careful Seeding.* Pages 1027–1035, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. ISBN: 9780898716245, Society for Industrial and Applied Mathematics, 2007.

[2] Xu Y., Qu W., Li Z., Ji C., Li Y., Wu Y. (2014) Fast Scalable k-means++ Algorithm with MapReduce. In: Sun X. et al. (eds) Algorithms and Architectures for Parallel Processing. ICA3PP 2014. Lecture Notes in Computer Science, vol 8631. Springer, Cham. https://doi.org/10.1007/978-3-319-11194-0_2