University of Groningen

Introduction to Data Science

Homework Assignment 6

Group 16:
Otte Tjepkema *(s3237184)*
José Rodrigues *(s4169328)*
Andrei Miculita *(s4161947)*
Robert Riesebos *(s3220672)*

October 16, 2019

Contents

1 Pen and paper question

$$\mathbf{D_0} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{array} \begin{array}{c} \begin{array}{ccccc} p1 & p2 & p3 & p4 & p5 \end{array} \\ \begin{bmatrix} 0.0 & 0.9 & 0.6 & 0.5 & 0.7 \\ 0.9 & 0.0 & 0.4 & 0.6 & 0.1 \\ 0.6 & 0.4 & 0.0 & 0.6 & 0.2 \\ 0.5 & 0.6 & 0.6 & 0.0 & 0.3 \\ 0.7 & 0.1 & 0.2 & 0.3 & 0.0 \end{bmatrix} \end{array}$$

1.1 Single linkage

We initially treat each point as a cluster. Then we repeatedly join clusters based on which of them have the minimum distance between any 2 points in them.

At the first iteration, the smallest distance between any 2 clusters in matrix D_0 is between $\{p2\}$ and $\{p5\}$, with value 0.1. We will therefore merge them and treat them as $\{p2, p5\}$. We then rewrite the distance matrix D_1, where $D_1(\{p2, p5\}, pn) = min(D_0(p2, pn), D_0(p5, pn))$, for $pn \in \{p1, p3, p4\}$. After that we symmetrically copy these distances on the column $\{p2, p5\}$, and the rest of the values remain unchanged.

$$D_1(\{p2, p5\}, p1) = min(D_0(p2, p1), D_0(p5, p1)) = min(0.9, 0.7) = 0.7$$
$$D_1(\{p2, p5\}, p3) = min(D_0(p2, p3), D_0(p5, p3)) = min(0.4, 0.2) = 0.2$$
$$D_1(\{p2, p5\}, p4) = min(D_0(p2, p4), D_0(p5, p4)) = min(0.6, 0.3) = 0.3$$

$$\mathbf{D_1} = \begin{array}{c} \\ p1 \\ \{p2, p5\} \\ p3 \\ p4 \end{array} \begin{array}{c} \begin{array}{cccc} p1 & \{p2, p5\} & p3 & p4 \end{array} \\ \begin{bmatrix} 0.0 & 0.7 & 0.6 & 0.5 \\ 0.7 & 0.0 & 0.2 & 0.3 \\ 0.6 & 0.2 & 0.0 & 0.6 \\ 0.5 & 0.3 & 0.6 & 0.0 \end{bmatrix} \end{array}$$

Next, we find the lowest distance between 2 clusters again: 0.2 between $\{p2, p5\}$ and $p3$. To build D_2 we need to find $D_2(\{\{p2, p5\}, p3\}, p1)$ and $D_2(\{\{p2, p5\}, p3\}, p4)$, so we apply the same formula:

$$D_2(\{\{p2, p5\}, p3\}, p1) = min(D_1(\{\{p2, p5\}, p1), D_0(p3, p1)) = min(0.7, 0.6) = 0.6$$
$$D_2(\{\{p2, p5\}, p3\}, p4) = min(D_1(\{\{p2, p5\}, p4), D_0(p3, p4)) = min(0.3, 0.6) = 0.3$$

And then build D_2:

$$\mathbf{D_2} = \begin{array}{c} \\ p1 \\ \{\{p2,p5\},p3\} \\ p4 \end{array} \begin{array}{c} \begin{array}{ccc} p1 & \{\{p2,p5\},p3\} & p4 \end{array} \\ \begin{bmatrix} 0.0 & 0.6 & 0.5 \\ 0.6 & 0.0 & 0.3 \\ 0.5 & 0.3 & 0.0 \end{bmatrix} \end{array}$$

Next we can build the last distance matrix D_3. We find the 2 closest clusters and merge them. In this case they are {{p2, p5}, p3} and {p4} with distance 0.3. Now we can find the distance between cluster {{{p2, p5}, p3}, p4} and {p1}:

$$D_3(\{\{\{p2,p5\},p3\},p4\},p1) = min(D_2(\{\{\{p2,p5\},p3\},p1), D_2(p4,p1)) = min(0.6,0.5) = 0.5$$

$$\mathbf{D_3} = \begin{array}{c} \\ \{\{\{p2,p5\},p3\},p4\} \\ p1 \end{array} \begin{array}{c} \begin{array}{cc} \{\{\{p2,p5\},p3\},p4\} & p1 \end{array} \\ \begin{bmatrix} 0.0 & 0.5 \\ 0.5 & 0.0 \end{bmatrix} \end{array}$$

Figure 1 shows the dendrogram we can obtain from single linkage clustering.

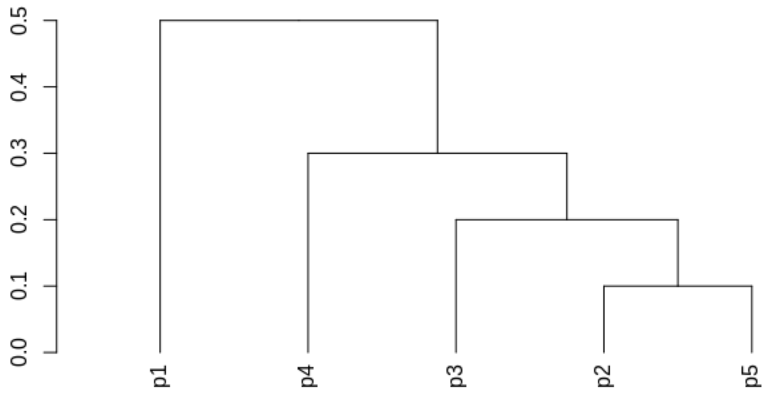


Fig. 1: Single linkage dendrogram

1.2 Average linkage

Similarly to single linkage, we initially treat each point as a cluster. Then we repeatedly join clusters based on which of them have the minimum average distance between all of their points.

At the first iteration, the smallest distance between any 2 clusters in matrix D_0 is between $\{p2\}$ and $\{p5\}$, with value 0.1. We will therefore merge them and treat them as $\{p2, p5\}$. We then rewrite the distance matrix D_1, where $D_1(\{p2, p5\}, pn) = avg(D_0(p2, pn), D_0(p5, pn))$, for $pn \in \{p1, p3, p4\}$. After that we symmetrically copy these distances on the column $\{p2, p5\}$, and the rest of the values remain unchanged.

$$D_1(\{p2, p5\}, p1) = avg(D_0(p2, p1), D_0(p5, p1)) = avg(0.9, 0.7) = 0.8$$
$$D_1(\{p2, p5\}, p3) = avg(D_0(p2, p3), D_0(p5, p3)) = avg(0.4, 0.2) = 0.3$$
$$D_1(\{p2, p5\}, p4) = avg(D_0(p2, p4), D_0(p5, p4)) = avg(0.6, 0.3) = 0.45$$

$$\mathbf{D_1} = \begin{array}{c c} & \begin{array}{cccc} p1 & \{p2,p5\} & p3 & p4 \end{array} \\ \begin{array}{c} p1 \\ \{p2,p5\} \\ p3 \\ p4 \end{array} & \begin{bmatrix} 0.0 & 0.8 & 0.6 & 0.5 \\ 0.8 & 0.0 & 0.3 & 0.45 \\ 0.6 & 0.3 & 0.0 & 0.6 \\ 0.5 & 0.45 & 0.6 & 0.0 \end{bmatrix} \end{array}$$

Next, we find the lowest distance between 2 clusters again: 0.3 between $\{p2, p5\}$ and $p3$. To build D_2 we need to find $D_2(\{\{p2, p5\}, p3\}, p1)$ and $D_2(\{\{p2, p5\}, p3\}, p4)$, so we apply the formula:

$$D_2(\{\{p2, p5\}, p3\}, p1) = avg(D_0(p2, p1), D_0(p5, p1), D_0(p3, p1)) = \\ min(0.9, 0.7, 0.6) = 0.7(3)$$
$$D_2(\{\{p2, p5\}, p3\}, p4) = avg(D_0(p2, p4), D_0(p5, p4), D_0(p3, p4)) = \\ min(0.6, 0.3, 0.6) = 0.5$$

And then build D_2:

$$\mathbf{D_2} = \begin{array}{c c} & \begin{array}{ccc} p1 & \{\{p2,p5\},p3\} & p4 \end{array} \\ \begin{array}{c} p1 \\ \{\{p2,p5\},p3\} \\ p4 \end{array} & \begin{bmatrix} 0.0 & 0.7(3) & 0.5 \\ 0.7(3) & 0.0 & 0.5 \\ 0.5 & 0.5 & 0.0 \end{bmatrix} \end{array}$$

Next we can build the last distance matrix D_3. We find the 2 closest clusters and merge them. In this case can be either {{p2, p5}, p3} and {p4} with distance 0.5, or {p1} and {p4} with the same distance. We randomly choose {p1} and {p4}, for the sake of variation, either one works. Now we can find the distance between cluster {{p2, p5}, p3} and {{p1},{p4}}:

$$D_3(\{\{p2,p5\},p3\},\{p1,p4\}) =$$
$$avg(D_0(p2,p1),D_0(p5,p1),D_0(p3,p1),D_0(p2,p4),D_0(p5,p4),D_0(p3,p4)) =$$
$$avg(0.9,0.7,0.6,0.6,0.3,0.6) = 0.61(6)$$

$$\mathbf{D_3} = \begin{array}{c} \\ \{\{p2,p5\},p3\} \\ \{p1,p4\} \end{array} \begin{array}{c} \begin{array}{cc} \{\{p2,p5\},p3\} & \{p1,p4\} \end{array} \\ \begin{bmatrix} 0.0 & 0.61(6) \\ 0.61(6) & 0.0 \end{bmatrix} \end{array}$$

Figure 2 shows the dendrogram we can obtain from average linkage clustering.

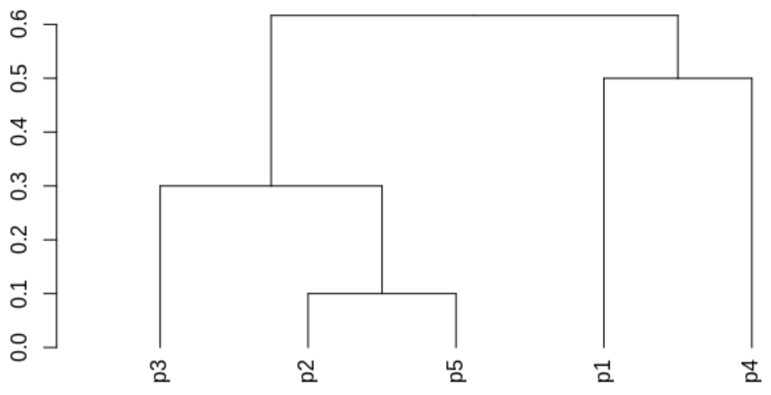


Fig. 2

Introduction

A galaxy is a large collection of of gas, dust and billions of stars that are bound by gravity. Larger galaxies can be formed by the merging of smaller ones. This merging usually only occurs between two galaxies, however multiple merging events can take place, which will create a larger and larger galaxy. When a merge occurs some fingerprint of the original galaxies still remains. In this report we will analyse a dataset which lists information about 280,000 stars and try to cluster them using their orbital parameters. More specifically two datasets will be used, one is the original dataset that has 6 dimensions, which correspond to the position x, y and z and velocities (v_x, v_y, v_z) of the stars. Another 3 dimensional dataset contains the energy, angular momentum, and angular momentum around the z-axis of the same stars. To perform this clustering a descriptive analysis was first performed. Firstly a spectral analysis was done to see what the intrinsic dimensionality of the data was. After that the size of the dataset was reduced by using sampling, to create 10 smaller datasets. Then K-means clustering was performed on the preprocessed data to find the galaxy clusters. The robustness of the K-means was also checked and it was also compared to Gaussian Mixture Model (GMM)

2 Descriptive analysis

2.1 The hint implies that the clusters are uniformly distributed within the data. This is useful information because it means that we can use random sampling of the dataset without introducing a bias towards certain clusters, providing we choose a large enough amount of samples.

2.2

2.3 In figure 3 the scree plot of the 6d data is given. Above each bar the variance explained by the principal component is given. To explain 95% of the data we need all principal components, since they each explain over 15% of the variance. This means that the intrinsic dimensionality is equal to the original dimensionality of the dataset. Therefore using PCA will not give any substantial benefit in our data processing.

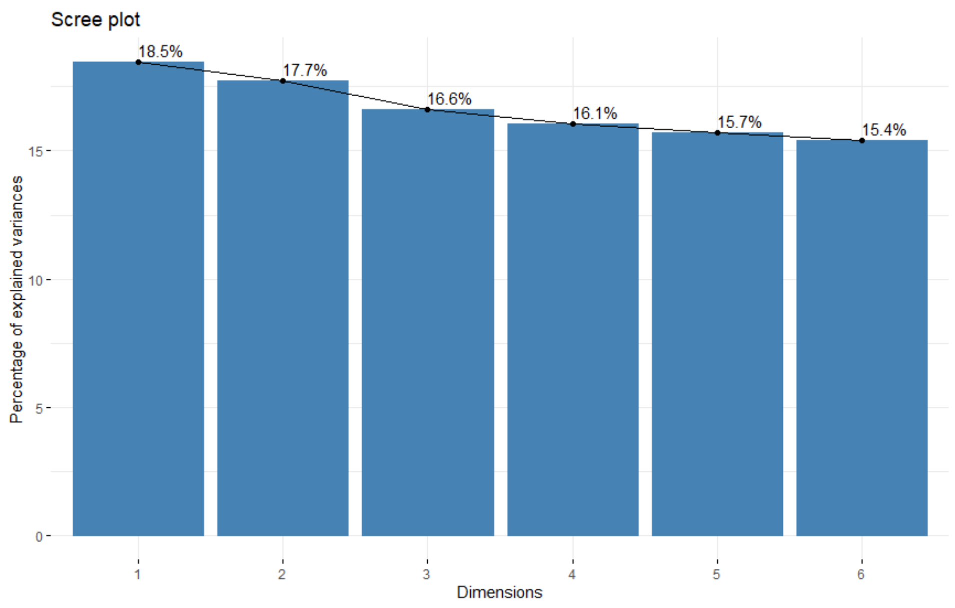


Fig. 3: Scree plot of 6d data

Next we look at the scree plot of the 3d data, which is shown in figure 4. Here we see that by taking the first two principal components we can explain 91.2% of the variance in the dataset. Hence we see that the intrinsic dimensionality of the dataset is closer to two. Therefore using PCA might help save computation time. However using PCA makes the results more difficult to interpret, this in combination with the fact that we lose 8.7% of the variance is a reason not to use it as a preprocessing method. In section 2.4.4 we will compare this preprocessing method with other methods to see decide whether we should use it.

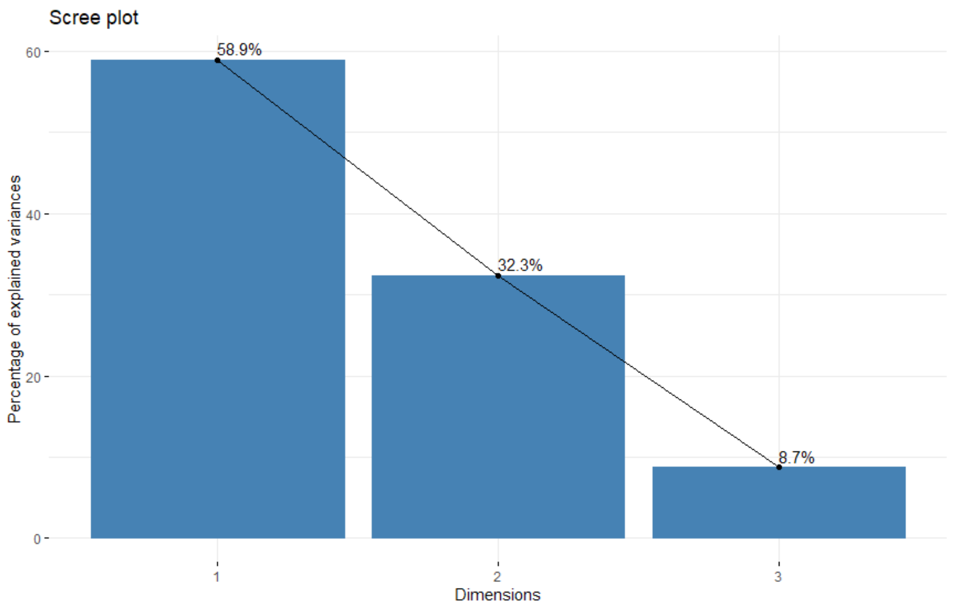


Fig. 4: Scree plot of 3d data

2.4 In this section we prepare training data sets for further analysis.

2.4.1 Since the datasets containing 3 and 6 dimensions each have 280,000 entries we need to use sampling to keep the computation times reasonable. The amount of samples that we have to use depends on two criteria. The first one is that we should have enough samples to accurately identify each cluster, the second one is that we should be able to comfortably handle the data with $\mathcal{O}(n^2)$ methods.

For K-means a general rule is that $5 * 2^k$ entries should be used for each cluster, where k is the number of dimensions [1]. Tables 1 and 2 show how many entries we would need to sample and the corresponding $\mathcal{O}(n^2)$ cost according to this rule for the 3d and 6d datasets.

Number of clusters	entries $(5 * 2^k)$	$\mathcal{O}(n^2)$ cost $\cdot(10^5)$
20	800	6.40
25	1000	10.0
30	1200	14.4

Tab. 1: Table showing the amount of entries and cost for different amounts of clusters for the 3d dataset. The amount of entries was found by multiplying the number of clusters by $(5 * 2^k)$

Number of clusters	entries $(5 * 2^k)$	$\mathcal{O}(n^2)$ cost $\cdot(10^6)$
20	6400	41.0
25	8000	64.0
30	9600	92.2

Tab. 2: Table showing the amount of entries and cost for different amounts of clusters for the 6d dataset. The amount of entries was found by multiplying the number of clusters by $(5 * 2^k)$

With the data from tables 1 and 2 we can derive how many samples we should use. We start with the 6d case, where in the case of 30 clusters, we would need on average 9600 entries. The cost of this is equal to $92.2 \cdot 10^6$ operations. We use the rough estimate that a modern computer can perform $5 \cdot 10^6$ operations per second, which was obtained by timing a simple for loop multiplying two random 5 digit numbers. If we would take 10 random subsets, an $\mathcal{O}(n^2)$ method on all of them would take roughly 180 seconds. This time is fairy reasonable, so more samples could be taken, since more samples will lead to better clustering. Therefore we will use 12000 samples for the 6d dataset, which should correspond to a computation time of around 5 minutes. This should also compensate for the fact that some of the clusters will have less than $5 * 2^k$ entries on average and will also allow us to take more clusters than 30 if necessary. For the 3d dataset we will take the same amount of entries, since the dimensionality of the dataset is less we would in theory need less entries for good clustering, but since the time requirement for processing this many entries is not that large we will take the large amount for better clustering. This will also allow us to take much more than 30 clusters if necessary. Using 10 sets of 12000 entries will give us a total of 120000 entries, which is

43% of the dataset. This also means that on average we have 43% of the samples of each cluster present in the sampled data. To randomly sample the data the 3d and 6d datasets were first shuffled. After which the datasets were divided into 10 groups each. Finally 12000 random samples were taken from each group. This technique was chosen over random sampling to make sure that no entry would appear in multiple subsets, which loses information from the full dataset.

To see the effects of the sampling we have made a table for both the 3d dataset with the mean, standard deviation and 3 largest eigenvalues from PCA, see table 1 and 2. Here we see that these properties of the full dataset and the subsets are very similar, hence we can conclude that our subsets are a good representation of the full dataset.

Type	Mean	Std. deviation	Eig 1 (%)	Eig 2 (%)	Eig 3 (%)
Full dataset	-35539	54162	58.95	32.35	8.70
Subset 1	-35391	53947	58.95	32.31	8.74
Subset 2	-35503	54090	58.86	32.43	8.70
Subset 3	-35299	53872	59.49	31.86	8.64
Subset 4	-35712	54389	58.96	32.28	8.77
Subset 5	-35409	54011	59.04	32.22	8.74
Subset 6	-35641	54267	58.68	32.59	8.72
Subset 7	-35502	54146	58.47	32.83	8.70
Subset 8	-35540	54132	59.24	32.12	8.64
Subset 9	-35700	54386	59.04	32.38	8.57
Subset 10	-35524	54122	58.91	32.31	8.78

Tab. 3: Table comparing the mean, standard deviation and percentage of variance for the 3 largest eigenvalues for the 3d dataset

Type	Mean	Std. deviation	Eig 1 (%)	Eig 2 (%)	Eig 3 (%)
Full dataset	0.7850	91.4661	18.47	17.74	16.61
Subset 1	0.6343	91.7456	18.80	17.57	16.70
Subset 2	0.6888	90.8680	18.35	17.89	16.55
Subset 3	0.9089	91.6548	18.16	17.87	16.70
Subset 4	0.6881	91.7844	18.80	17.73	16.73
Subset 5	1.1066	91.1129	18.57	17.74	16.51
Subset 6	0.7387	90.7172	18.35	17.85	16.68
Subset 7	0.8374	91.2938	18.42	17.71	16.60
Subset 8	0.9633	91.9798	18.44	18.01	16.81
Subset 9	0.9487	91.1070	18.61	17.91	16.56
Subset 10	0.8433	91.3797	18.04	17.75	16.60

Tab. 4: Table comparing the mean, standard deviation and percentage of variance for the 3 largest eigenvalues for the 6d dataset

2.4.2 Next we analyse the clustering tendencies of the data. We do this by means of the Hopkins statistic. We use the 1-H definition where the further the value of the Hopkins statistics is under 0.5, the better the data will lend itself to clustering. We will also use two types of preprocessing, z-score tranformation and PCA to see if this affects the clustering tendency of the dataset, the result of this can be seen in table 5. Note that these results only show how well the dataset would cluster, it says nothing about the quality of the obtained clusters, and hence we will only use it for exploratory analysis. We see that the 3d dataset seems to be better at clustering for every type of preprocessing. This makes sense since the data present in the 3d dataset is created from cleverly transformed data from the 6d dataset. This reduces the dimensionality of the dataset which makes it better at clustering. Furthermore we see that Z-score preprocessing has the lowest Hopkin statistic, which is also logical since it normalizes distances between points which is better for clustering. We will check these result again in section 3 to see if the Hopkins method was correct.

Preprocessing	3d dataset	6d dataset
None	0.06973749	0.158035
Z-score	0.02550292	0.0941237
PCA	0.02768404	0.09602955

Tab. 5: Table showing the Hopkins statistic for the 3d and 6d dataset with different kinds of preprocessing, a lower value means a higher cluster tendency

3 K-Means Clustering

3.1 In an attempt to check how many clusters the data may contain, a K-Means Clustering method was used for different numbers of clusters. This way we could check the squared average distance between the data points and the cluster centers (inertia) and choose a number of clusters that would minimize it (firstly without accounting for computation time constraints), and using a range of k[20,40] clusters, the optimal number was 39.

After a Random selection of points from a prior Random selection of subsets, 10 different subsets of 12000 samples each were selected for further analysis (this number optimizes the computation time without sacrificing the results). For these subsets we also plotted the sum of squared distances vs the number of means, k, for k in range 20 to 100. Again $k = 39$ seemed like a fairly good elbow point.

Using K-Means Clustering, each data point was assigned to a cluster. In order to visualize the scattering of data, Principal Component analysis was made in order to transform the dataset into a 2D matrix that could be easily plotted. The cluster assignment process results are represented below:

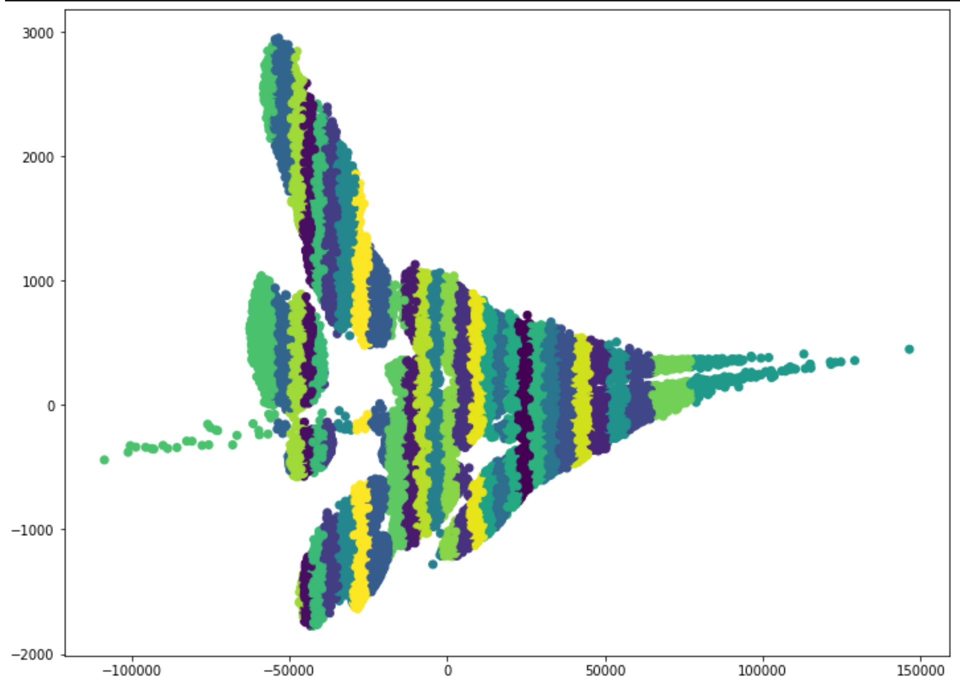


Fig. 5: Dataset points Clustering Representation

3.2 In order to evaluate the quality of the evaluation method depicted in 3.1, the squared average distance to the cluster centroid was chosen as a metric.

For comparison purposes and to guarantee a solid evaluation of the different models, the ellbow method was used. This way, by finding the point of inflexion of the inertia curve, one can obtain a pretty good guess of the number of clusters a certain dataset might have.

For every one of the 10 subsets, the mean squared distance to the centroids and the standard deviation (from the "ellbow" method results) were computed both for the 3D and 6D cases.

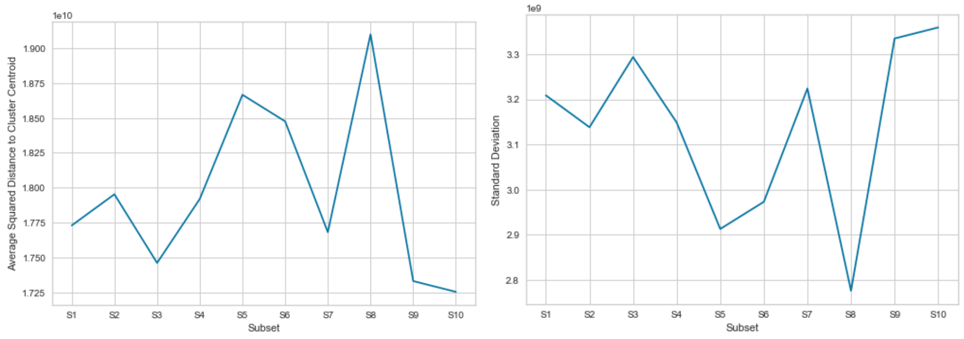


Fig. 6: Average Squared Distance to Cluster Centroid and Standard Deviation for the 3D subsets

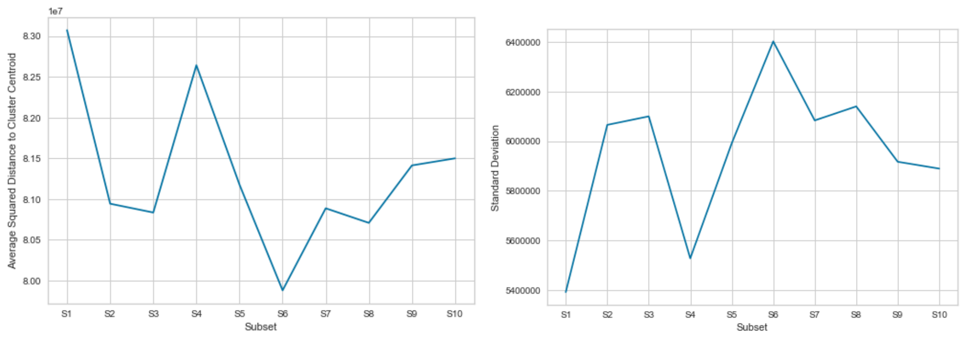


Fig. 7: Average Squared Distance to Cluster Centroid and Standard Deviation for the 6D subsets

Even though the Standard Deviation (STD) values seem too big, when compared to the ”ellbow” method distortion score (same as inertia), one can observe that the STD is actually one order of magnitude smaller than the average distortion score value.

If we compare the relative error between the ”ellbow” method's distortion score and the 10-subset model K-Mean's inertia value, they are actually pretty close. That shows that a 39 cluster guess is not too far from optimal. This is represented in the graph bellow.

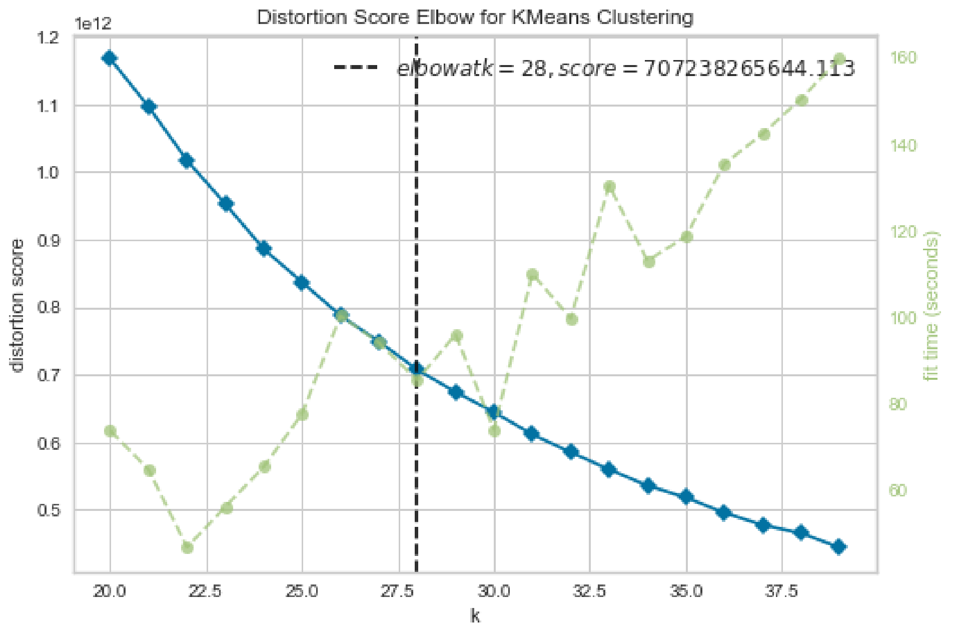


Fig. 8: "Ellbow" Method Distortion Score

By calculating the point of inflexion of the distortion score curve, the optimal number of clusters for this dataset is estimated to be 28. Comparing the results in the graph for 28 clusters with the earlier guess of 39 clusters, it can be concluded that despite the average centroid distance for both cases is smaller for the 39 clusters, the choice of 28 clusters is actually better. One can see as well that after the k=33, the computation time grows really fast, so it may not be a good idea to choose too large values of k (computationally-wise).

3.3 Finally to check the robustness of the k-means method we try to see if the 10 K-means models obtained from the 10 subsets, assign the same points to the same cluster. This test was performed on the 3d dataset since it has the lowest number of dimensions and therefore the fastest computation time. If the clustering is not robust for the 3d, it is most likely not robust for the 6d dataset either, because of the higher dimensionality. Ideally we would like to test this on the full dataset, however due to its size this would take an extremely long amount of time. Therefore we create a smaller subset of the data to test this on. We create this subset by picking out 10 points that are closest to the center of each cluster of each K-mean model. Since we take 10 points from 39 clusters for 10 models this turns out to be a total of 3900 points. The random initialization of the K-means method means that the same clusters in the different K-mean models have different numbers. This was corrected for by giving the clusters from the different K-mean models where the centers were closest to each other the same number.

Table 6 shows the results of this experiment. We see that 34.9 % of the points were put in the same cluster in all the K-means models. We also see that a large amount of points only have 1 or 2 wrongly clustered points. However there are still also a substantial amount of points where the different K-means models assigned data points to different clusters. The conclusion that we can pull for this is that the K-means models are reasonably robust. This is also what we would expect. Since we did not perform multiple initializations in the training of the K-mean models, because of time constraints, it is likely that they are stuck in a local optima, instead of being closer to the global optimum. Also each model was trained on only a small subset of the full dataset, which means that even if they all converged to the global optimum there still would be small differences in the position of the clusters. Therefore to get more robust K-means models more points would have to be taken, as well as taking more initializations in the training of the models.

Wrongly clustered points	Frequency	Percentage
0	1362	34.9
1	704	18.1
2	665	17.1
3	457	11.7
4	429	11.0
5	279	7.15
6	4	0.10

Tab. 6: Table showing the frequency of wrongly clustered points for the 10 K-means model. The percentage of the frequency in comparison with the total number of points is also shown.

4 Gaussian Mixture Model

4.1 We have a Gaussian Mixture Model (GMM) defined by:

$$p(x) = \sum_k w_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

$$\mathcal{N}(\mu_k, \Sigma_k) = \frac{\exp(-0.5(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k))}{\sqrt{|2\pi\Sigma_k|}}$$

Where k is the number of Gaussians, w_k is the mixing coefficient for Gaussian k and $\mathcal{N}(x|\mu_k, \Sigma_k)$ is the Gaussian k with corresponding mean μ_k and covariance matrix Σ_k.

In order to find the unknown (Gaussian) parameters $\theta = (w, \{\mu_k\}, \{\Sigma_k\})$ that maximize the likelihood, we use an Estimation Maximization (EM) approach, of which the pseudo-code is listed below.

First the Gaussian parameters are initialized. This can be done randomly or, for a better outcome, previous results of a k-means algorithm can be uses, which is done in the pseudocode.

After initialization the Estimation Maximization algorithm is executed. The EM algorithm consists of two main steps that are repeated for each iteration, or so-called epoch; the Expectation (E) step and the Maximization (M) step. In the E step we compute, for each sample x_i, the probability that it belongs to Gaussian k. This is called the responsibility. The responsibility of Gaussian k for sample x_i is given by:

$$\gamma_{ik}^{(m)} = \frac{w_k^{(m)} \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{l=1}^{k} w_l^{(m)} \mathcal{N}(x_i|\mu_l, \Sigma_l)}$$

Next, in the M step, we use the computed responsibilities to adjust the Gaussian parameters for each Gaussian k. The new parameters are calculated as follows:

$$\begin{aligned} w_k^{(m+1)} &= \frac{N_k}{N} \\ \mu_k^{(m+1)} &= \frac{1}{N_k} \sum_n \gamma_{nk} x_n \\ \Sigma_k^{(m+1)} &= \frac{1}{N_k} \sum_n \gamma_{nk} (x_n - \mu_k^{(m+1)})(x_n - \mu_k^{(m+1)})^\top \end{aligned}$$

where $N_k = \sum\limits_n \gamma_{nk}$ and N is the total number of samples.

Finally we evaluate the log likelihood after each iteration. The log likelihood is given by:

$$\log p(x|\theta) = \sum_n \log(\sum_k w_k \mathcal{N}(x_n|\mu_k, \Sigma_k))$$

When the log likelihood no longer changes execution of the algorithm stops.

Algorithm 1 Pseudo-code of EM Algorithm for determining the parameters of a GMM model

```
$N \leftarrow$ number of samples
$K \leftarrow$ number of clusters
$\mu[0..K] \leftarrow$ centroids obtained from $k$ means
$\Sigma[0..K] \leftarrow$ identity matrices
$w[0..K] \leftarrow$ samples in cluster $k$ / $N$
$likelihood \leftarrow \infty$
$last\_likelihood \leftarrow -\infty$

while $likelihood - last\_likelihood > threshold$ do
  for $k$ in $1..N$ do
    for $x_i$ in $data$ do
      $\gamma[N \times K] \leftarrow \frac{w_k \mathcal{N}(x_i|\mu_k,\Sigma_k)}{\sum\limits_{l=1}^{k} w_l \mathcal{N}(x_i|\mu_l,\Sigma_l)}$
    end for
  end for

  for $\gamma_{nk}$ in $\gamma$ do
    $N_k \leftarrow \sum\limits_{N} \gamma_{nk}$
    $w_k \leftarrow \frac{N_k}{N}$

    for $x_n$ in $data$ do
      $\mu_k \leftarrow \frac{1}{N_k} \sum\limits_{N} \gamma_{nk} x_n$
      $\Sigma_k \leftarrow \frac{1}{N_k} \sum\limits_{N} \gamma_{nk}(x_n - \mu_k)(x_n - \mu_k)^\top$
    end for
  end for
  $last\_likelihood \leftarrow likelihood$
  $likelihood \leftarrow \sum\limits_{N} \log(\sum\limits_{K} w_k \mathcal{N}(x_n|\mu_k, \Sigma_k))$
end while
```

4.2 We use the first sample subset of both the 3d and 6d dataset to perform k means clustering. After k means is done we use the obtained means as initial means for the GMM EM algorithm. We clearly see that there are clusters of stars with higher and lower energy, and also clusters at the ‘tips’ where the angular momentum is highest.

4.3 To compare the likelihood for different number of clusters we plot the (log) likelihood for k = 20 to 40, shown in figure 9. For each value of k the EM algorithm is run until either 100 iterations are reached, or the increase in likelihood is less than 1e-3.

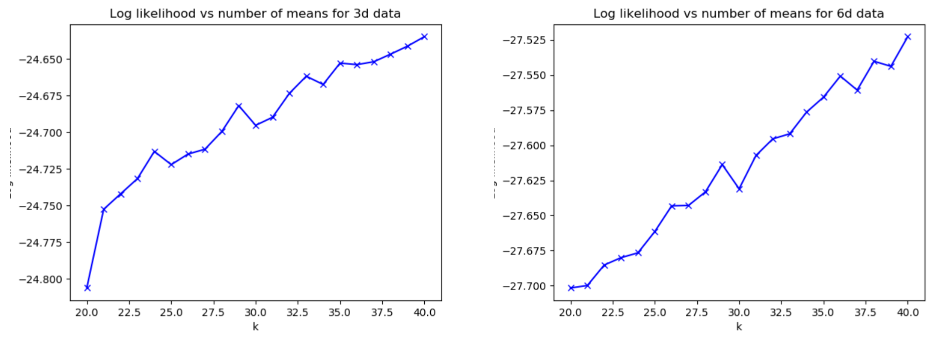


Fig. 9: Likelihood plotted for k = 20 to 40, for 3d and 6d data

We observe that for both the three and six dimensional data the likelihood seems to go up as the number of clusters increases, which implies that more clusters results in a better GMM model for both datasets. Furthermore the likelihood for the 6d data is lower than the likelihood of the 6d data. For both k means and GMM we chose the best of 10 random initializations. As we see in the plots in figure 11b, a higher likelihood doesn't imply that the clustering is better. When we plot the likelihood for k = 20 to 100 (but with only 1 random initialization due to time constraints) we observe that the likelihood still seems to increase if the number of clusters increases.

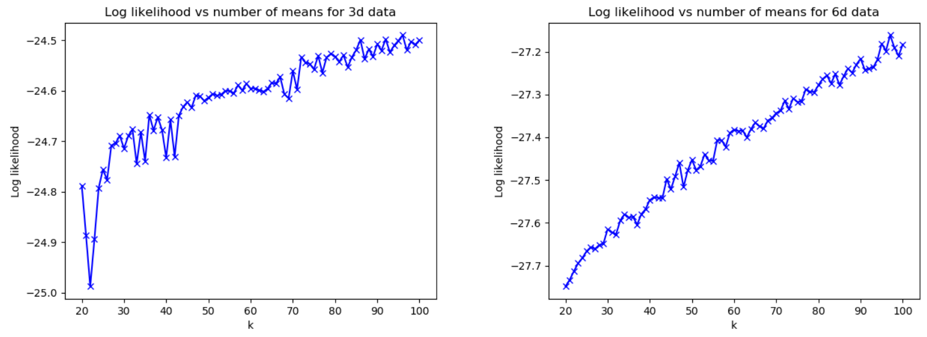
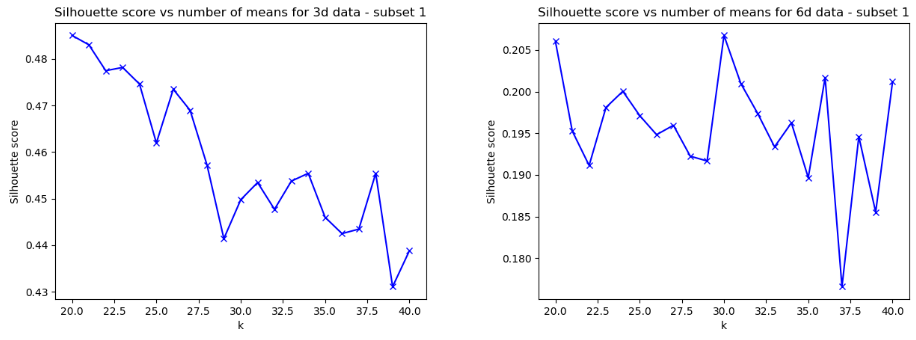
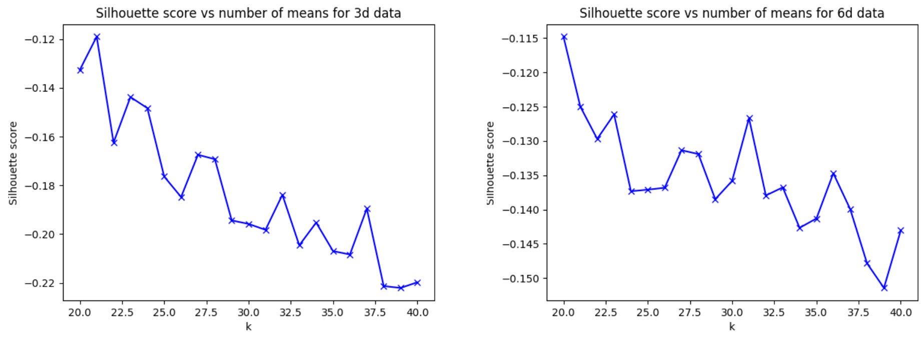


Fig. 10: Likelihood plotted for k = 20 to 100, for 3d and 6d data

To compare k means with GMM we calculate and graph the silhouette scores for both the 3d and 6d datasets for k from 20 to 40, shown in 11b (for computation time's sake up to 40 means are used). The silhouette scores of GMM are based upon labels obtained by classifying each sample as belonging to the cluster with the highest probability. We observe that the silhouette scores for k means are significantly higher than the scores for GMM. This indicates that a better clustering is achieved when using k means (where 'better' implies that the clusters are denser and better separated).



(a) Graphs for k means



(b) Graphs for GMM

Fig. 6: Silhouette scores for k means and GMM

5 Bonus challenges

5.1

5.2 To get an initial feel for the clustering predictions we plot the points of the 3d data and color them according to the predicted clusters. The result is shown in figure 12 and 13.

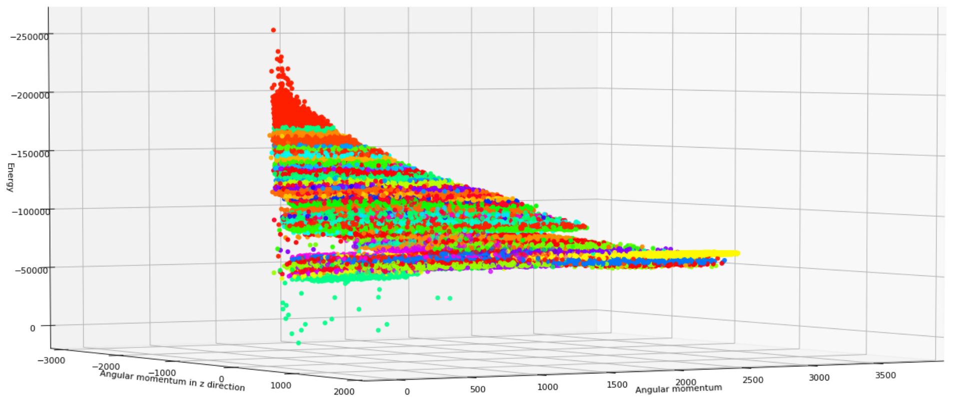


Fig. 12: 3d points plotted and colored according to predictions

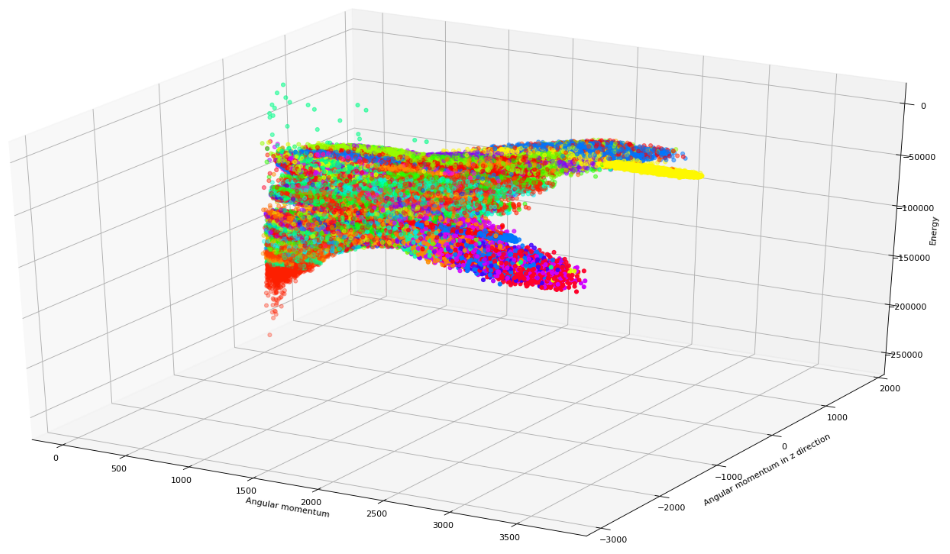


Fig. 13: 3d points plotted and colored according to predictions

To investigate which clusters of stars could be part of a galaxy which most recently merged and which stars might be part of a galaxy of a very early merge we calculated the sum of squared distances for each cluster and divided these sums by the number of stars in the cluster.

This yielded a measure of distance to the cluster center, which is useful because galaxies of early mergers are more diffuse. The results are shown in table 7. The table is ordered from clusters that are part of galaxies of a very early merge to clusters that are part of galaxies wherein stars most recently merged.

cluster number	Sum of squares
26	120546.6758298878
32	131506.50423023614
35	134299.18098470318
28	140548.91746480673
16	142177.26723129186
37	142197.4562719001
33	144218.19858450515
5	144734.07775106787
21	144927.5533669939
34	145121.35190394567
30	146222.21044844255
20	147682.29012289748
31	152225.0627098246
0	153148.90666927947
14	153379.10768452415
11	153714.07043780736
1	154147.0309759449
8	154934.4109007002
2	158156.16344560953
18	160678.76152643177
24	161040.1931768813
15	161871.44319987672
9	162255.0521914178
19	167313.7068709147
23	168045.14910013368
17	169387.5575739852
38	169784.94673351734
3	171427.00353043806
10	171449.73232303007
6	171556.68752400368
22	172174.7906017712
12	174468.22820270958
4	175449.33673229092
13	180154.6111527371
25	181649.33012400457
7	184926.55955711298

Tab. 7: Sum of squares for each cluster

A Sum of squared distance plots

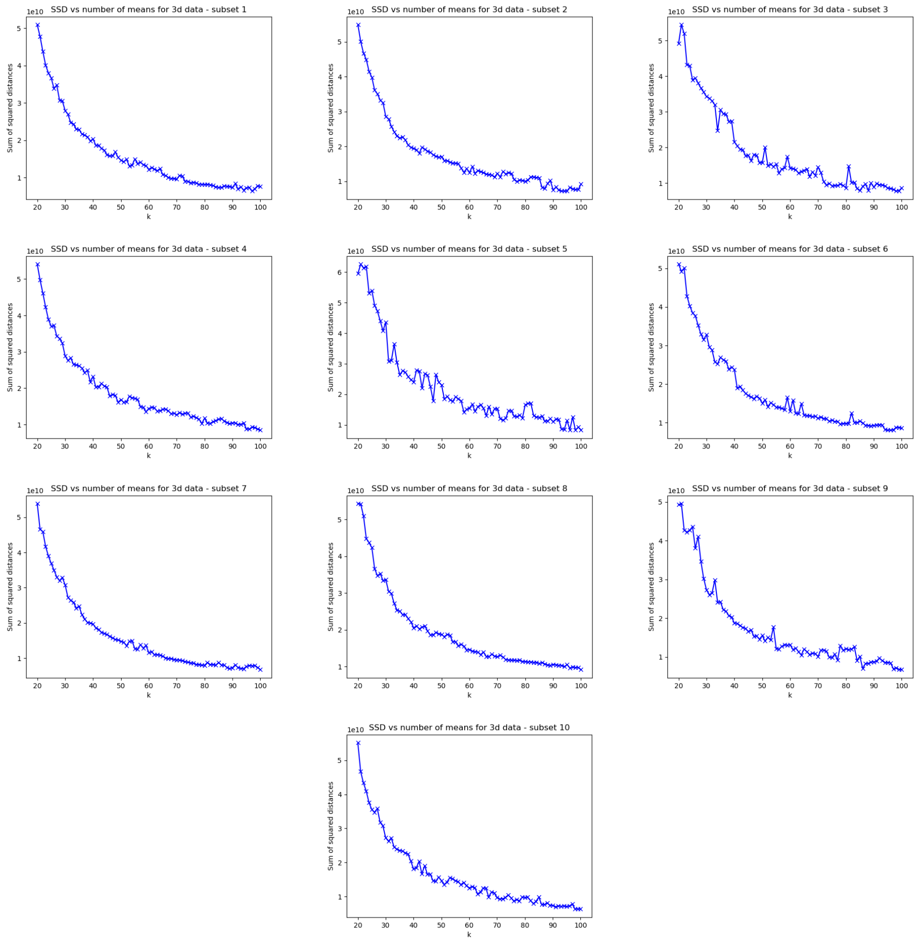


Fig. 14: Sum of squared distances plotted for k = 20 to 100 of 3d data

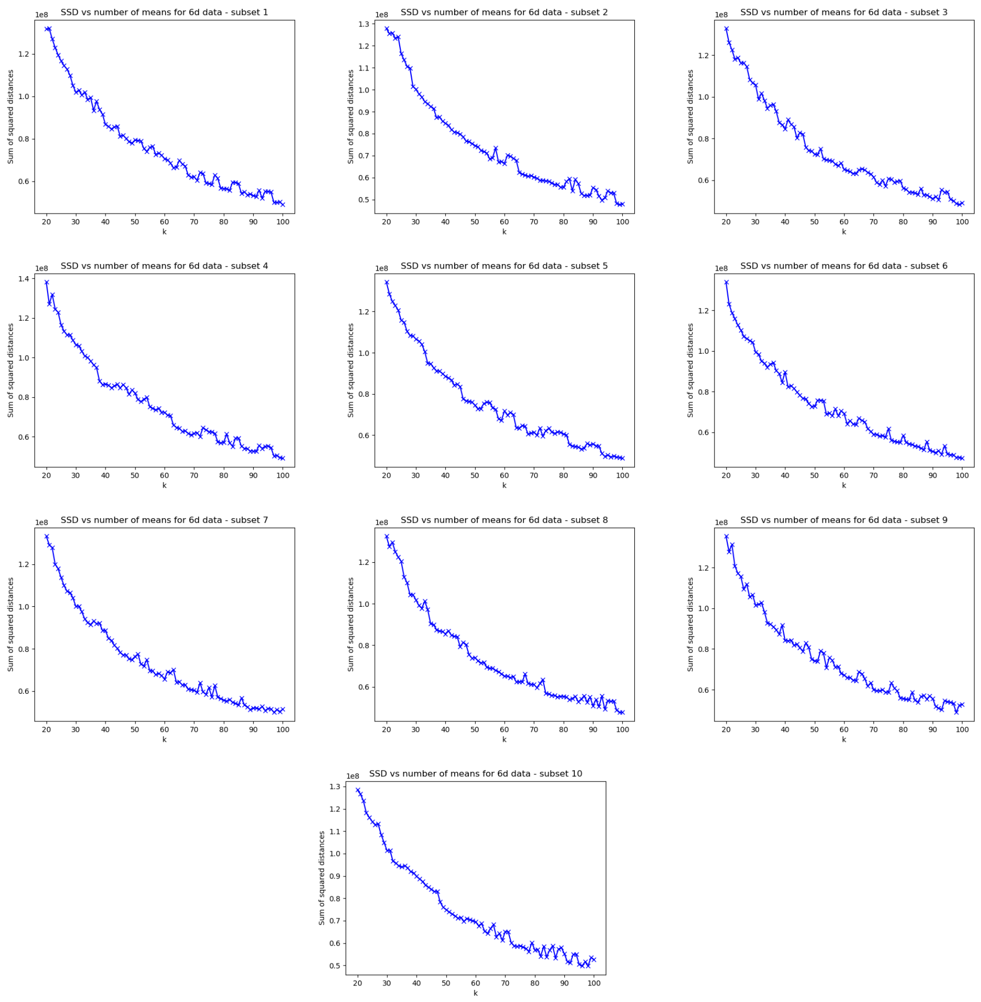


Fig. 15: Sum of squared distances plotted for k = 20 to 100 of 6d data

Bibliography

[1] Sara Dolnicar. *A Review of Unquestioned Standards in Using Cluster Analysis for Data-Driven Market Segmentation.* University of Wollongong: Research Online, 2002.