

# Data Mining CA4010

*Sian Lennon 16343896*

*Shanon Mulgrew 16304263*

A report submitted to Dublin City University, School of Computing for module CA4009: Search Technologies, 2019/2020. I understand that the University regards breaches of academic integrity and plagiarism as grave and serious. I have read and understood the DCU Academic Integrity and Plagiarism Policy. I accept the penalties that may be imposed should I engage in practice or practices that breach this policy. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations, paraphrasing, discussion of ideas from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references. I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work. By signing this form or by submitting this material online I confirm that this assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study. By signing this form or by submitting material for assessment online I confirm that I have read and understood DCU Academic Integrity and Plagiarism Policy (available at: <http://www.dcu.ie/registry/examinations/index.shtml>)

Name(s): Sian Lennon, Shannon Mulgrew

<b>1.Introduction</b>	<b>3</b>
<b>2. Dataset Description</b>	<b>3</b>
2.1 Dataset	3
2.2 Goal	4
<b>3.Data Preparation/Cleaning</b>	<b>4</b>
3.1 Formatting dataset	4
3.2 Attribute selection	6
3.3 Workshop & discoveries	8
<b>4. Clustering and Algorithm selection</b>	<b>9</b>
4.1 Standardization	9
4.1.1 Continuous float standardization	9
4.2 Clustering algorithm	10
4.2.1 K-Means	10
4.2.2 The Python Code	12
<b>5. Experiments and Analysis</b>	<b>14</b>
5.1 Discounting the 'mass' attribute	14
5.2 Changing values of k to find the optimum cluster value	17
5.3 Getting rid of the categorical attributes	18
5.4 Picking less random values for initial centroids	19
5.5 Classifying the test data with our new centroids	20
5.6 Additional findings	21
<b>6. Conclusion</b>	<b>22</b>

# 1.Introduction

---

The following report outlines our investigation and use of the kaggle dataset of Meteorite landings. In this report we used many techniques and data mining skills that we learned in the CA4010 module. Our report shows our insights into the dataset clustering in general. The report below outlines the steps we took to prepare, standardize and cluster our dataset.

## 2.Dataset Description

### 2.1 Dataset

---

The dataset we based our prediction on was called “Meteorite Landings”. The dataset was found on kaggle. It contained roughly 45,000 entries, each entry being a meteor that hit earth from outer space. The original dataset had 10 attributes containing information on each meteor. These attributes consisted of **name**, **id**, **nametype** (valid or relict), **reclass** (class type of meteor, over 12577 unique types ), **mass** (kg), **fall** (if it fell or was found), **year**, **reclat** (latitude), **reclong** (longitude) and **Geolocation** (lat,long). For this assignment we omitted some attributes, however this will be discussed later on.

The choice to use this dataset for our assignment was made based mostly on the usefulness of the dataset itself. With 45,000 entries, we thought the size of the dataset was an appropriate size for the assignment. We also thought the idea of working with both numerical and categorical data would be very insightful. In addition to this, we noticed that the null values within the dataset were not too frequent and not many rows would need to be omitted.

Link for dataset - <https://www.kaggle.com/nasa/meteorite-landings>

## 2.2 Goal

---

The original goal of the assignment was to predict the location of a meteor to hit earth based on the other attributes given such as mass, fall, year and class. We planned to investigate the relationship between meteors with different attribute values and their corresponding Geolocation.

After the initial analysis of our dataset from the data preparation, we thought that clustering the data would be a very interesting approach. We came about this approach due to the fact that many entries in our data seemed to be entered at the same time, meaning, geolocation, reclat and reclang were the same, with the same or similar masses and the only different input was the name. After discovering this we decided to approach our data with a clustering algorithm.

We decided to cluster our data into groups of similar objects which are dissimilar to objects in other clusters. In doing this, we hoped to show that the meteors in the dataset can be grouped displaying that there is a clear pattern of behaviour found in this data. This was our hypothesis from our data preparation. We also wanted to find the optimum amount of clusters to sort our dataset into.

## 3.Data Preparation/Cleaning

---

The dataset we got was from kaggle, this did not mean however that the data was read to use as it was. The data needed to be prepped for analysis before we could begin to do our experiments on it. We used many techniques for formatting and cleaning the dataset.

### 3.1 Formatting dataset

---

Some initial preparation we did was to format the data into an appropriate form for analysis. Due to the amount of entries in our dataset it made more sense to remove null values rather than replace them with a global constant of 'unknown'.

Our first step was completely removing all rows where null values were found in any column.

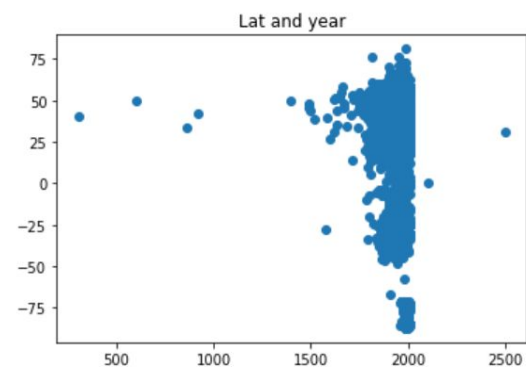
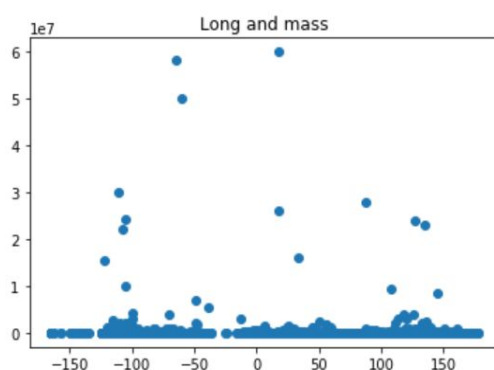
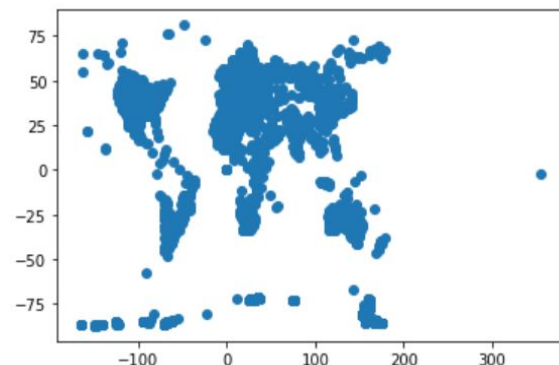
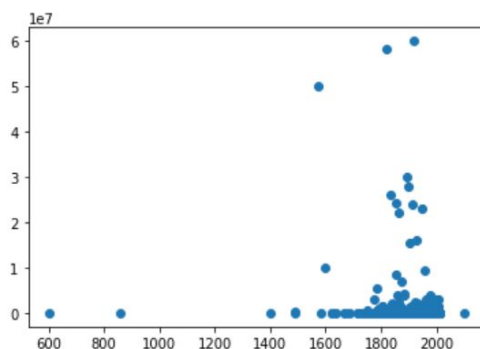
```
newdata = data.dropna()
```

We also decided to drop duplicate values within the dataset to prevent us dealing with duplicate entries where all columns including 'name' attribute were the same.

```
def drop_dup(data):  
    data=data.drop_duplicates()  
    return data
```

From previous work for our presentations, we came across the fact that we may have some extreme outliers. As our approach was to cluster our data which can be sensitive to outliers we decided to trim the data in order to remove these outlying points.

To visualize these outliers in regards to certain columns we plotted them in a scatter plot. (left= mass, year) (right lat, long)



As you can see there are many obvious outliers present in our data that we'll have to take into consideration when applying our algorithms. These could be from misinterpreted data, human error or unknown reasons.

## 3.2 Attribute selection

---

The next step in the process was to look at the attributes of our dataset, deciding on what could and would be used for our model.

Our attributes:

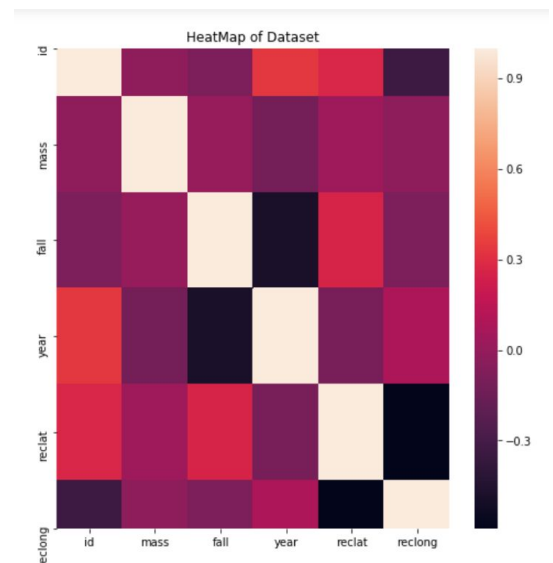
- name: the name of the meteorite (typically a location, often modified with a number, year, composition, etc)
- id: a unique identifier for the meteorite
- nametype: one of:
  - *valid*: a typical meteorite
  - *relict*: a meteorite that has been highly degraded by weather on Earth
- recclass: the class of the meteorite; one of a large number of classes based on physical, chemical, and other characteristics (see the Wikipedia article on meteorite classification for a primer)
- mass: the mass of the meteorite, in grams
- fall: whether the meteorite was seen falling, or was discovered after its impact; one of:
  - *Fell*: the meteorite's fall was observed
  - *Found*: the meteorite's fall was not observed
- year: the year the meteorite fell, or the year it was found (depending on the value of fell)
- reclat: the latitude of the meteorite's landing
- reclang: the longitude of the meteorite's landing
- GeoLocation: a parentheses-enclose, comma-separated tuple that combines reclat and reclang

As 'name' and 'id' were unique values for each meteor, these two attributes were omitted first. Next, 'reclat' & 'reclang' were omitted as they were not needed with the presence of 'GeoLocation'. The last attribute completely removed and not needed for the prediction process was 'nametype', we originally planned on using this attribute however upon inspection, we found that every entry had the value 'Valid'. See the

Valid	100%
Relict	0%

'drop\_unused\_attributes()' function for details on how the columns were deleted.

We decided to plot a heat map to see if there was any insight into the correlation between our attributes. As you can see with the key attributes such as fall and reclat seem to have some correlation along with fall and reclang and mass and reclat, reclang and year.



From studying our dataset and the heat map we decided to remove some attributes using the function below.

The dropped values:

```
#drop unwanted attributes
def drop_unused_attributes(data):
    del_attributes = ['nametype', 'name', 'id', 'GeoLocation']
    data = data.drop(columns = del_attributes)
    return(data)
```

We were then left with five attributes, these were:

- *Reclass*
- *Mass*
- *Fall*
- *Year*
- *Reclat*
- *Reclang*

After all this was done we split our dataset into a training set (80%) and test set (20%). The reason for this will be discussed further later. Our analysis was done totally with our training set. This will be our new dataset for the clustering.

### 3.3 Workshop & discoveries

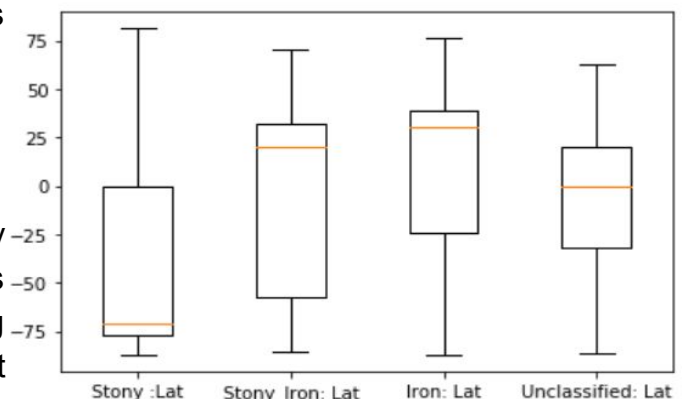
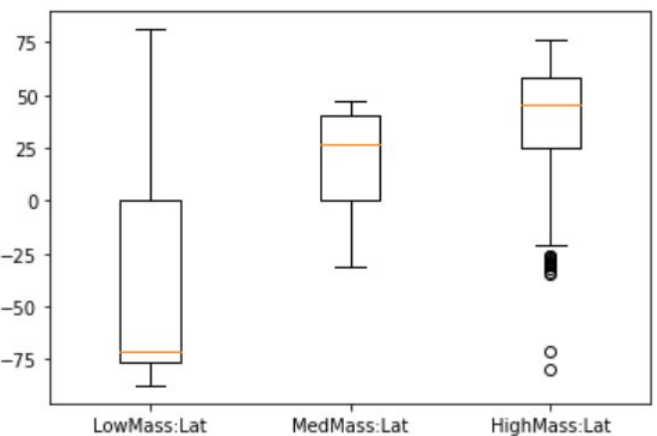
---

This section will go through our workshop and analysis of our findings from it.

The workshop we chose was data dispersion. In hindsight, our workshop was not as insightful as it should have been. The dispersion of data is a very important indication of where to go with your analysis. However, looking over the graphs after the workshop we did make some discoveries that concluded to us creating the clustering algorithm instead of the original prediction idea.

After re-reading through our workshop presentation and taking into account the feedback given we noticed that there were many data points grouped together in the same areas in regards to class, mass, fall and year. We could see in our boxplots that much of our data was congregating around the lower hemisphere for low mass, stony class and year around 1980. However, the data seemed to congregate around the upper hemisphere for high mass, Stony\_Iron class and year above 2000. The graphs beside us show this distribution.

This got us thinking that there could possibly be clusters of similar data where all attributes are very similar. We decided that a clustering approach would possibly give us more insight into the data we're dealing with rather than any other approach.





## 4. Clustering and Algorithm selection

---

When starting this project, the idea of regression or clustering came to mind as we were predicting a continuous attribute and our dataset did not suit classification as it was. Knowing that our data was in fact labeled and not unsupervised, it seemed like regression was our most suited algorithm. However, from our data dispersion workshop, we decided that clustering the data would be an interesting approach. That is when we decided to use our data as unlabeled data and perform a clustering analysis on the dataset. This would enable us to separate the data into different clusters and find patterns of behavior with the meteors in doing so.

### 4.1 Standardization

---

After data preparation, there were a few initial steps to set up our data for clustering. This involved standardizing the data so extra weight was not given to different attributes when substituting in the euclidean distance formula. For example, we wanted to add no extra weight to the years attribute than the mass attribute, however, the years attribute had a greater range and mean than mass did. Standardizing the attributes removed this extra weight and allowed an even playing field for all attributes.

#### 4.1.1 Continuous float standardization

---

The Mean Absolute Deviation and the Mean were found for each attribute. Mean absolute deviation was used instead of the standard deviation to reduce the effect of outliers as the deviation from the mean is not squared as in standard deviation. This was used to find the z score for each attribute value which was used in our clustering algorithm. Z score was found using the z-score algorithm  $Z = (x - m) / s$ . Where x is the attribute value, m is the attribute mean and s is the mean absolute deviation.

$$z = (\text{point} - \text{mean}) / \text{mad}$$

## 4.2 Clustering algorithm

---

### 4.2.1 K-Means

---

The clustering algorithm chosen at first was the k-means algorithm. The standardization file as seen above describes the process of standardizing the attributes so that an accurate distance measurement could be found. The clustering file goes through the process of splitting the data into clusters. Where each cluster contains items that are similar to the other items in the cluster and dissimilar to the items in the other clusters.

1. The k-means algorithm starts with defining a number, k for the number of clusters we wish to define. Our first iteration started with four clusters. We randomly selected four centroids from our dataset.
2. Using these four meteor entries as our centroids we could split the dataset into the four clusters. We did this by sending in each dataset entry and each centroid into the euclidean distance formula created by us. The Euclidean distance formula took the z-score of each entry and centroid attribute and subtracted them from each other. This was then squared. This was done for all numerical attribute values. The categorical value and the binary value were compared to each other as strings and given a 1 for a match and a 0 for a no match.
3. Once the distance from each point to each centroid was found, the smallest distance from each point to a centroid indicated the centroid each point was closest to and therefore the centroid it belonged to. The data set was then split up into these clusters.
4. From there, new centroids were computed by finding the mean point of each cluster. This was done by adding up all the attribute values in each cluster and dividing by the number of entries in that cluster. Since there was no way to get the categorical data mean, we decided to use the mode of the categorical in each cluster of the data in the new centroid. I.e what was the most frequent value

for reclass and fall in each cluster than became the new centroid value for that cluster.

5. After the new centroids were computed, we repeated steps 2-3. After step 3 was complete, we needed to see if the clusters had changed at all or stayed the same. If they stayed the same, the process was complete. So we compared our new clusters to the old clusters from the last iteration. If they were the same, we moved on to step 6. If they were not we repeated this step until they were.
6. Step 6 involved finding the error function or 'objective' function of the clusters. This was to see the tightness of the clusters. I.e if they were spread out or condensed. The lower the number the better. We got this by summing the squares of the euclidean distance of each entry in each cluster to their respective centroids.
7. The last step was printing the centroids, length of the clusters and the objective function for analysis into our results file.

## 4.2.2 The Python Code

---

*Euclidean distance formula:*

```
def euclidean_distance(centroid, entry):

    # Compute distance between two points.
    sums = []
    # append categorical distance to sums i.e fall, class
    reclass = compare_categorical(entry[0], centroid[0])
    fall = compare_categorical(entry[1], centroid[1])

    sums.append(reclass)
    sums.append(fall)

    i = 2
    while i < len(entry):
        # Get z score of the object at attribute i and centroid
        # at attribute i.

        entry_z = stnd.z_score(float(entry[i]), i-1)
        if i == 2:
            print(entry[i])
            print(entry_z)

        centroid_z = stnd.z_score(float(centroid[i]), i-1)

        # Euclidean distance (x-y)^2
        tmp = (entry_z-centroid_z)**2
        # Append to sums list
        sums.append(tmp)

        # Increment
        i+=1

    total = 0

    # Total sums list for Euclidean distance
    for s in sums:
        total = total + s

    if total < 0:
        total = total*-1
    # Get square root of total as last part of formula
    euclidean = math.sqrt(total)

    return(euclidean)
```

*Re-evaluating the centroid:*

```
def re_evaluate_centroid():  
  
    #Find mean of new clusters  
  
    newCentroids = []  
  
    #print(len(clusters))  
    for cluster in clusters:  
        if len(cluster) != 0:  
  
            i = 0 # Attribute  
  
            obj = cluster[i]  
            centroid = []  
  
            while i < len(obj):  
                total = 0  
                j = 0 # Object  
  
                new_list = []  
                while j < len(cluster):  
                    if i < 2:  
                        new_list.append(cluster[j][i])  
                        j+=1  
                    else:  
                        total = total + cluster[j][i]  
                        j +=1  
                if i < 2:  
                    mean = most_frequent(new_list)  
                else:  
                    mean = total/(len(cluster))  
  
                centroid.append(mean)  
  
                i +=1  
  
            newCentroids.append(centroid)  
    return(newCentroids)
```

*Objective function:*

```
def objective_function(centroids):  
    print("Calculating objective function.")  
    total = 0  
    i = 0  
  
    while i < len(clusters):  
        for entry in clusters[i]:  
            total = total + (euclidean_distance(centroids[i], entry))**2  
        i+=1  
    return(total)
```

## 5. Experiments and Analysis

---

After we completed the code that clusters the data, and finds the objective value of the clusters, we began our experiments. In order to display the results, we printed the centroids, lengths of the clusters, objective and the number of iterations it took to cluster the data all gathered in a to a text file. This made it easier to compare the data from our experiments.

The initial trial run of the data after the algorithm was completed was not the result we expected. Because of this, we changed part of our algorithm to see why the clusters appeared different than what we originally expected.

### 5.1 Discounting the 'mass' attribute

---

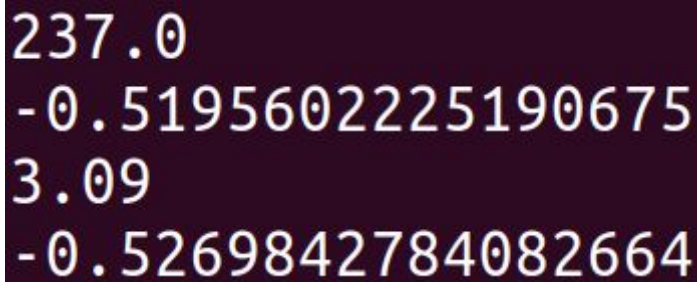
After our initial experiment of the clustering algorithm run over all the attributes, we found that the clusters did not match the analysis of our data. When looking into the reason for this, we noticed the centroids of many of the clusters had a very high mass. This section will go over how we came to this conclusion and what we did to counteract this.

### Results and analysis:

The initial clustering (where  $k=4$ ) saw that there was a very uneven balance of clusters. Two clusters had ~16,000 entries assigned to them. And the other two had  $< 10$ . The objective of this experiment was 114703.39240225336. What was not expected from this were the uneven clusters seen. The two centroids with the bulk of the data clustered around them were:

1. ['L6', 0, 11730.307702644808, 1996.9136684933683, 5.426745732666144, -5.785840853149065]
2. ['L6', 0, 1072.822303096355, 1985.0642745361056, -71.16995488380265, 108.83425958187078]

No matter how many times we ran the program results did not differ from this behaviour. Considering that a lot of our mass, seen from our data dispersion workshop was below 20kg, the mass column (3rd item from left) of the centroids is very high. This is when we thought that the high-value outliers of mass in our dataset could be causing the new centroids mass to be higher than it should have been. To prove this, we printed the value of mass for some of the entries over the z-score for these to get some insight into where we were going wrong. Here is an example:



```
237.0
-0.5195602225190675
3.09
-0.5269842784082664
```

After this was printed, we noticed that even though in reality there was a huge difference in the mass of these two dataset objects, there was a very slight difference in the z-score of them. We knew then that this column was throwing off our analysis.

After we realised that mass was causing this we decided to experiment with the mass column discounted in our analysis. This resulted in much more condensed and even clusters. Here is the results of this experiment:



```
-----  
Results for Clustering dataset where k = 4. And Mass attribute discounted.  
-----
```

```
Centroids = [['L6', 'Found', 1893.0712025316457, 28.103491362341796,  
-17.193632450949384], ['L6', 'Found', 2000.9796640141467, -8.94131552637787,  
-30.61477985956393], ['L6', 'Found', 1989.9204093567253, -75.52459460748504,  
112.55910863250482], ['H6', 'Found', 1998.7570653191092, 20.90543476268015,  
33.54191968294973]]
```

```
Length of each centroid:
```

```
Cluster 1 = 1264
```

```
Cluster 2 = 6786
```

```
Cluster 3 = 17100
```

```
Cluster 4 = 5343
```

```
Objective = 88156.84586420542
```

```
Number of iterations before fully clustered = 19
```

See above that the objective (error function) is much lower than when mass is counted. As well as this, the dataset objects are more spread out over the clusters. This proved further that the mass column was what was throwing off our analysis.

Even after trimming our data to remove outliers, the k-means algorithm did not work well for the mass attribute. This was down to the range and mean of this attribute being so high even though the bulk of the data belonged in such a small range of 0kg - 20kg. So when the new centroids were being computed, the mean of mass was found for each centroid, the objects for high-value mass were creating a high mean causing disruption in accuracy in our algorithm.

After this discovery was made, we had a few choices. We could trim the data further to discount the data with an outlying high mass, we could try a different method (most likely k-medoids) to reduce the impact of outliers or we could discount the mass attribute in our k-means algorithm. As we had trimmed our dataset and already cut out a good portion of our dataset in the data preparation stage, we did not want to reduce it further. As for picking a more robust method, we decided this was not the way to go as the outliers were so extreme that even k-medoids could be skewed from the attribute. From here, we decided to discount the mass column in our analysis.



## 5.2 Changing values of k to find the optimum cluster value

---

This section will go through the process for finding the optimum value for k, the number of clusters. We did this by looking at the objective error function for experiments where k was changed. The objective shows how closely clustered together the clusters are so the lower the function the better the clusters. Naturally, the function will reduce as the number of clusters increases. However, the trick here was finding a value for k where the objective is significantly lower than the objective for k-1 and not much higher than the objective for k+1.

### Results and analysis:

We computed the clusters and objective error function value for k in range 2-8 inclusive to find the 'correct' value for k. Where correct refers to the most useful value for k.

The results of these experiments showed that the optimum value for clusters seems to be k = 5. The value for objective error function decreases greatly from 88156.84586420542 for k=4 to 64663.60610634702 for k=5, a difference of ~24000. Whereas, it only reduces ~10000 for k=3 to k=4 and ~3000 from k=5 to k=6.

Here are the results for k=5:

```
-----
Results for Clustering dataset where k = 5. And Mass attribute discounted.
-----

Centroids = [['L6', 'Fell', 1847.3822937625755, 30.723573515090525,
-6.250401301810863], ['L6', 'Found', 1993.7038371313672,
-77.48196589209265, 146.32998204222278], ['L6', 'Found', 2002.740698120445,
11.853286020233957, 7.56736650163024], ['L6', 'Found', 1937.0619539316917,
27.4433247561557, -31.123312524225565], ['H4', 'Found', 1983.3907108112348,
-71.16395223081743, 14.32200271724516]]

Length of each centroid:
Cluster 1 = 497
Cluster 2 = 11936
Cluster 3 = 10428
Cluster 4 = 1259
Cluster 5 = 6373

Objective = 64663.60610634702

Number of iterations before fully clustered = 22
```

## 5.3 Testing without the categorical attributes

---

We did notice after the last set of experiments that 4 out of 5 of the centroids were of class L6 and all contain the value 'found' in the 'fall' attribute. As we couldn't take an actual mean of the categorical attributes, we were taking the mode instead. This made us wonder if this was screwing our results at all. So for the next experiment, we decided to discount the categorical variables and see if we get the same results for just the numerical values. Obviously, if we did get the exact same results or something very similar, it would seem our algorithm was not working as expected as we want our results to take into account these attributes also. For this experiment, we computed for  $k=4$  and  $k=5$  with no categorical attributes.

### Results and analysis:

What we saw from these experiments was that the objective function was significantly reduced when the categorical attributes were not counted in the analysis.

For  $k=5$ , the objective function was 64663.60610634702 with categorical attributes counted and 46223.82678262171 when they were not counted. Obviously, a slight reduction in the objective function could be expected as there are fewer attributes to compare however, this is a significant reduction.

Our analysis of this is that as there are so many different values for reclass and that could be causing the objective function to be so high. We know from our data dispersion workshop that there are 12577 different values in 'reclass'. Since the mean of the reclass attribute cannot be computed, we took the mode. That means that all values that are not the mode will not be the same and will be computed as a distance of 1. If we had a way for the granularity of the values to not be so low, as in to put these values into sections, that would be more useful for this analysis.

Another factor is the binary attribute 'fall', we know that all values of fell will be computed as a distance of 1 for all the centroids for  $k=5$  as they all contain the value 'found'.

At the end of the day, there is no point in ruling out the categorical values just because they bring the objective function up so they will still be counted.

## 5.4 Picking less random values for initial centroids

---

When doing the previous experiments, we noticed that for each run of the program the results differed slightly, even when  $k$  was the same value. We thought that this could be down to picking a 'random' value for the original centroids. We thought it would be beneficial instead to handpick the original values for these centroids to see if we could get the objective function down.

The centroids we picked for this were based off the trend seen in the data from the data dispersion section. We chose:

```
[[L6, 'Found', 1980, -95.00, 50 ],[L6, 'Fell', 2000, 40, 0 ],['H4', 'Found', 1975, -51.00, 23.00], ['L6', 'Found', 1877, 28.00, -8.00]]
```

We chose these centroids from the dataset as they seemed like the points that could represent the initial clusters we saw in our data dispersion analysis.

### Results and analysis:

Obviously, as we had only seen four possible clusters in our initial analysis of the data, we could only compute  $k=4$ . So we compared our new results where our initial centroids were hand-chosen to our old results for the random centroids.

The result for this shows that the objective function decreased by quite a substantial amount ~12,500 from 88,000 to 75,500 when these clusters were picked. Less surprisingly, the number of iterations of the re-calculation of the clusters went down from 22 to 13. This was expected as the initial centroids **should** be closer to the final centroids. This result confirmed for us that the clusters we had seen in the original analysis of our data were actually close to the clusters computed by our algorithm.\

## 5.5 Classifying the test data with our new centroids

---

After we fully computed our final clusters and centroids, we thought the next step would be to investigate clustering unseen data. We took the test set we had previously split in the data preparation section and sorted it into clusters based on our newly computed centroids. We then found the objective of this and were looking to see how it compares to running the test set over the k-means algorithm.

### Results and analysis:

This was probably the best result for us which proved that the clustering algorithm worked. Here are the results, the top is the algorithm ran from scratch on the test data, below that is the classifying of the test data based on the previously computed centroids found using the training set.

```
-----  
Results for Clustering Test-Set where k = 5. And Mass attribute discounted.  
-----  
  
Centroids = [['L6', 'Found', 2002.3837076513132, 11.747042747240169,  
7.273557373429772], ['L6', 'Found', 1885.4163568773233, 29.446828513011138,  
-17.48150661710037], ['L6', 'Found', 1987.470211402947, -76.9360042024343,  
158.8454885522104], ['L5', 'Found', 2000.6820113314448, -77.01364510269126,  
130.27884939518407], ['H4', 'Found', 1981.6921322690991, -65.10079564424177,  
9.133806352907639]]  
  
Length of each centroid:  
Cluster 1 = 2627  
Cluster 2 = 269  
Cluster 3 = 1561  
Cluster 4 = 1412  
Cluster 5 = 1754  
  
Objective = 24624.14200972857  
  
Number of iterations before fully clustered = 14
```

```
-----  
Results for Classifying Test-Set where k = 5 .And centroids are predetermined  
-----
```

```
Centroids = [['L6', 'Fell', 1847.3822937625755, 30.723573515090525,  
-6.250401301810863], ['L6', 'Found', 2002.740698120445, 11.853286020233957,  
7.56736650163024], ['L6', 'Found', 1993.7038371313672, -77.48196589209265,  
146.32998204222278], ['H4', 'Found', 1983.3907108112348, -71.16395223081743,  
14.32200271724516], ['L6', 'Found', 1937.0619539316917, 27.4433247561557,  
-31.123312524225565]]
```

```
Length of each centroid:
```

```
Cluster 1 = 119
```

```
Cluster 2 = 2605
```

```
Cluster 3 = 2959
```

```
Cluster 4 = 1645
```

```
Cluster 5 = 295
```

```
Objective = 22639.995730099778
```

```
Number of iterations before fully clustered = 1
```

The above shows that classifying the data using the previously computed centroids from our training set resulted in a lower objective function. This was a good result as it showed that new data could be classified with our model and will result in an accurate classification.

## 5.6 Additional findings

---

Throughout our analysis phase we saw some commonalities for all experiments to do with our algorithm and the dataset. This section will go through these findings.

- For all runs of the algorithm, the final centroids were very spread apart which is a good result and what is expected.
- When running the algorithm over our data for the same value of  $k$ , it was interesting to see that the results of objective and final centroids differed only very slightly, meaning that few entries were in different clusters. The difference was down to the different values of initial centroids picked. However, even though objective was the same, number of iterations differed a lot for the same values of  $k$ . This was down to the randomness of the initial centroids and how long it took to reach the final centroids.

- The results prove our hypothesis of a pattern of behaviour present in the dataset that shows clusters of similar data. If we were to understand why this is the case for meteors, we could conclude that entries were made in bulk when a number of similar meteors were found/seen in the same place. This could be due to excavations on areas or meteor showers. Of course this is just speculation as we can't be sure.

Final clusters for k=5:

```
[[['L6', 'Fell', 1847.3822937625755, 30.723573515090525, -6.250401301810863], ['L6', 'Found', 1993.7038371313672, -77.48196589209265, 146.32998204222278], ['L6', 'Found', 2002.740698120445, 11.853286020233957, 7.56736650163024], ['L6', 'Found', 1937.0619539316917, 27.4433247561557, -31.123312524225565], ['H4', 'Found', 1983.3907108112348, -71.16395223081743, 14.32200271724516]]
```

## 6. Conclusion

---

To conclude, this data mining assignment was very insightful for understanding clustering and how it works. The project overall helped us grasp the concepts of data dispersion, standardization, and k-means. As well as this, the analysis of the results allowed us to gain insights into the dataset and clustering in general and its usefulness/flaws. In hindsight, the dataset we used was not a very well suited dataset to run a clustering algorithm over, especially k-means. This was because of the categorical attributes and the high large range of the mass attribute. Of course, the k-medoids algorithm is more robust to outliers so this would have been a better route to take. But, even with all the road-blocks taken into account, we did manage to cluster our data using k-means and after tweaking the algorithm to suit our dataset, we are happy with the results we achieved.

External sources for research:

- <https://www.statisticshowto.datasciencecentral.com/dispersion/>
- <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a>
- <https://medium.com/towards-artificial-intelligence/get-the-optimal-k-in-k-means-clustering-d45b5b8a4315>

