CA270 Clustering Project - Joseph & Adam

Section 1: Dataset Description

We found this dataset in the PyCaret library

Name: Employee.csv

Description: This dataset records the features of ~15,000 employees at an unnamed company. It consists of a mix of both categorical and numerical variables.

Attributes:

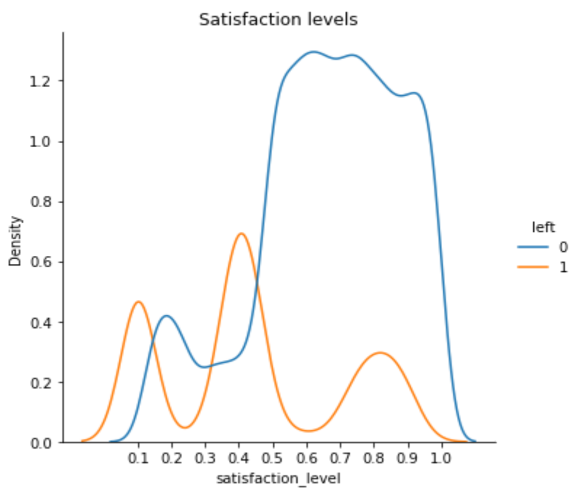
- **satisfaction_level:** Satisfaction levels for employees in range [0, 1]
- **last_evaluation:** Most recent work performance evaluation in range [0, 1]
- **number_project:** Number of projects employee is involved in
- **average_monthly_hours** Average amount of hours an employee does a month
- **time_spend_company:** Number of years spent at the company
- **Work_accident:** Suffered from a work accident. 1: True, 0: False
- **promotion_last_5_year:** Promoted in the last 5 years. 1: True, 0: False
- **department** The department to which an employee belongs
- **salary:** categorical [low, medium, high]
- **left:** Left the workplace. 1: True, 0: False

As can be seen, the dataset does contain both categorical and numerical attributes. In this project, we will attempt to form multiple clusters of employees and class them according to the "left" class. A cluster of tuples will be labelled 1 if this group of employees are predicted to leave, and 0 if they are believed to stay.

We will be using both k-means clustering algorithm and an agglomerative hierarchical clustering algorithm. We believe that using these will result in some differences which we can compare to get an idea of which algorithm seems more useful in this context.

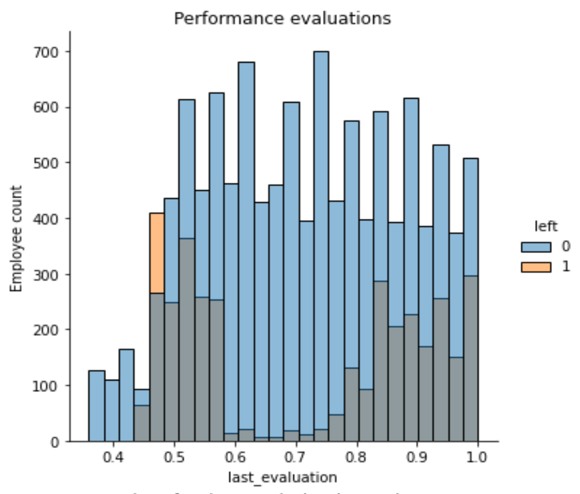
Section 2A: Analysis of the Dataset - Joseph

Distribution of attributes in relation to class

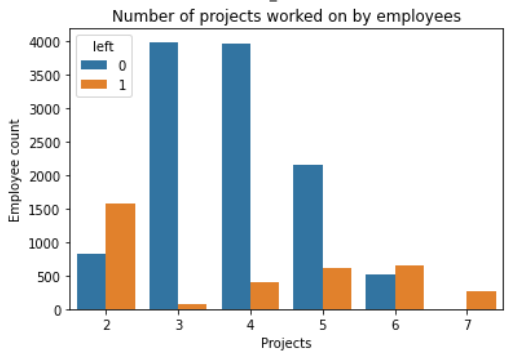


(i) The vast majority of employees showed satisfaction levels greater than 50%. There is a spike at the low end of satisfaction reports.

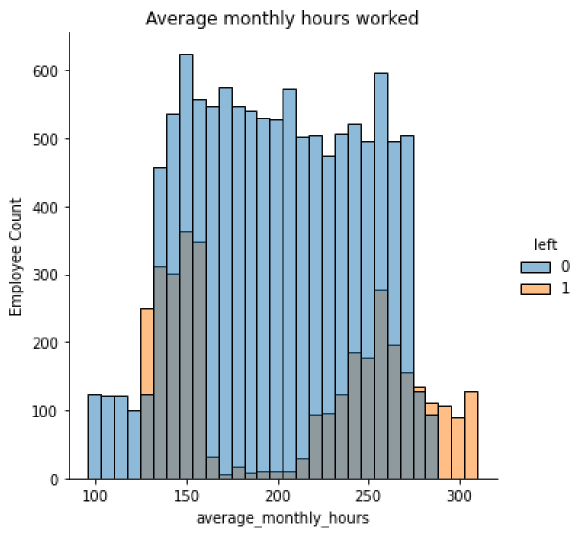
A large proportion of employees who left had a satisfaction level lower than 0.5. Besides this, a considerable proportion of employees with low satisfaction levels (>0.3) stayed, and some employees with high satisfaction left.



(ii) The distribution of performance evaluations show that employees with median performance report figures don't tend to leave the company. This attribute may contribute greatly to the generation of the clusters.

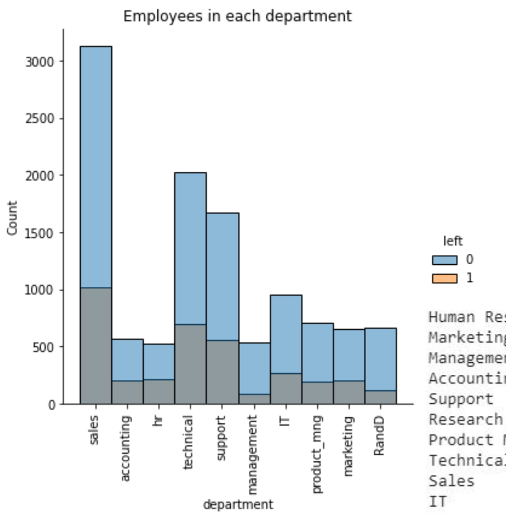


(iii) For employees working on 2, 6, and 7 projects we observed that the number of employees who left the company are more than those who stayed at the company. In fact, employees put on 7 projects all left the company! This is a strong statistic that will certainly be a driver for clustering.



(iv) Here we see a bimodal distribution with a high number of employees with hours around 150 and hours around 250. We see that most employees that left are employees with work hours around these modes. i.e. not around the median. Also, employees with significantly high work hours (> 275) show higher tendency to quit.

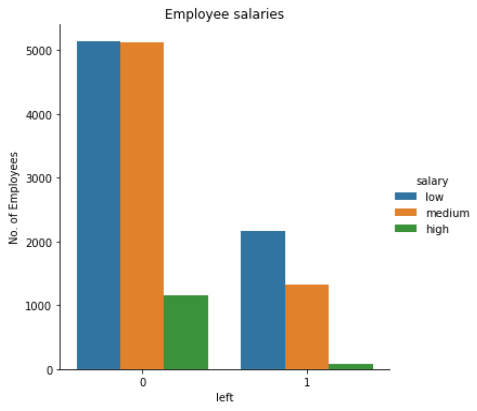
Section 2B: Analysis of the Dataset - Adam



```
                                prop
Human Resources             0.290934
Marketing                   0.236597
Management                  0.144444
Accounting                  0.265971
Support                     0.248991
Research & Development      0.153748
Product Management          0.219512
Technical                   0.256250
Sales                       0.244928
IT                          0.222494
```

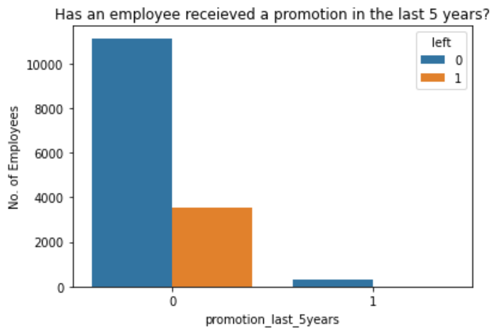
(v) Department - As can be seen in the graph, there is an even spread of people leaving the company across all departments with regards to the proportion of people in a department. This shows that no department is much worse than the others to work for. It can be seen also that the people in management have a lower proportion of people leaving.

The proportions however show that hr loses the most staff as a proportion of the total staff in that department with 30% leaving. Only 14% of management left. As there isn't a huge spread between departments I don't believe departments will be a huge help in classifying clusters.

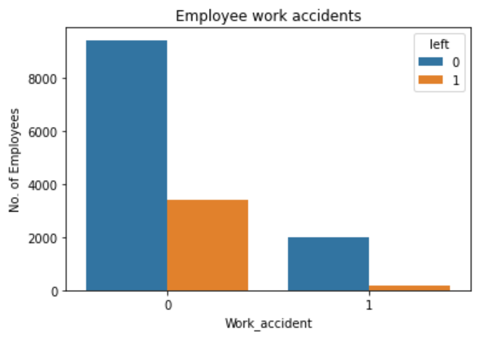


(vi) Salaries - It can be seen in the dataset that in the majority of the departments there are a lot more people with a low salary, which you would expect. In management however, the majority of staff have a high or medium salary.

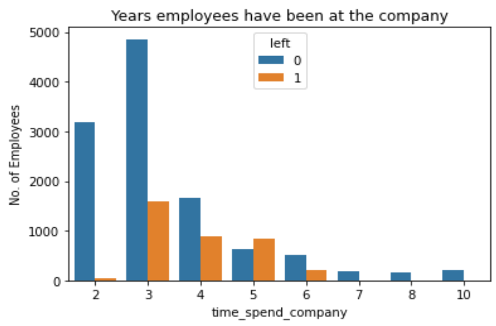
In the graph you can see that very few people with a high salary left, the majority were people on low wages. This may prove useful later.



(vii) Promotion - Promotions seem to be a rare thing in this company. Of the people who were lucky enough to receive one, only a couple left. It can be seen by comparing the 2 sets of bars that the likelihood of an employee leaving having gotten a promotion was much lower than if they hadn't.



(viii)Work Accident - Whether or not an employee had an accident or not didn't impact an employee staying or leaving. If anything the graph shows that the opposite is true. A smaller percentage of the people who had a work accident left the company than those that had no work accident. This may be as a result of the copious amounts of compensation they may have received as a result.



(ix) Years Employed - This graph shows that any employee that has been working at this company for 7+ years will most likely not leave. It seems an employee will usually work for 3 years at least before quitting, this should be useful for our clustering algorithms.

Section 3: Describing the algorithms

Joseph: Agglomerative Hierarchical Clustering.

Distance measures

Firstly I will discuss distance measures that I will use for the implementation. There are mostly quantitative variables with which Manhattan distance would work. There are categorical variables which will preferably use another distance measure. I will use a function which will compute **individual variable pairwise distance matrices** and get the average of the pairwise distances across the matrices as a final distance matrix. (**Gower distance**) For quantitative variables, **normalised Manhattan** and for categorical variables, **Dice similarity coefficient**.
Note that I am **OneHotEncoding** categorical variables before getting their Dice similarities. As a consequence of this, the similarity between categorical variables is either 0 or 1. Also, $\sqrt{(xi - xj)^2} = |xi - xj|$ so using euclidean distance would be equivalent in this case. The distance measure formulas are as follows:

Normalised Manhattan (for individual variable):

$\frac{|xi - xj|}{range(x)}$ where xi and xj are values of the variable

Dice similarity::

$\frac{2|Xi \cap Xj|}{|Xi| + |Xj|}$ where Xi and Xj are sets containing value of variable xi and xj

Algorithm

We start with each object in a cluster of its own and then repeatedly merge the closest pair of clusters until we end up with just one cluster containing everything. The process can be visualized on a **dendrogram**.

1. Assign each object to its own single-object cluster. Calculate the distance between each pair of clusters.
2. Choose the closest pair of clusters and merge them into a single cluster, reducing the total number of clusters by one.
3. Calculate the distance between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all the objects are in a single cluster.

There are several optimisations which cause the algorithm to run faster. For example, since dist(row i, row j) = dist(row j, row i), only one side about the diagonal of the distance matrix needs to be computed. Also, the distance between two clusters is simply the shortest distance between the clusters. I.e. the distance between two points in the clusters which are closest to each other. (**single-link clustering**)

Adam: K-Means Clustering.

Distance measures

For my K-Means clustering algorithm I intend to use **Euclidean Distance** to compute the distance between the tuples. I have converted the salary column into numerical form with low = 1, medium = 2 and high = 3. I have decided to drop the departments column as from my analysis I can see it is not crucial and there is no way to quantify the difference between sales and hr. For this reason I think **OneHotEncoding** these values will lead to them having a bigger impact on the data then they seem to from analysis. The formula for computing Euclidean Distance is as follows.

$$\sqrt{(xi - xj)^2 + \dots (yi - yj)^2}$$

Where x and y are attributes and xi is the attribute x in the ith tuple. j corresponds to the centroid from which the distance is being computed.

Before using the Euclidean Distance formula however I plan to standardize the tuples using the typical **Standardization Formula** below.

$$\frac{x - \mu}{\sigma}$$

Where μ is the mean of the column and σ is the standard deviation.

In this way the values of all attributes will be unitless and so will not skew the results of the clustering algorithm.

Algorithm

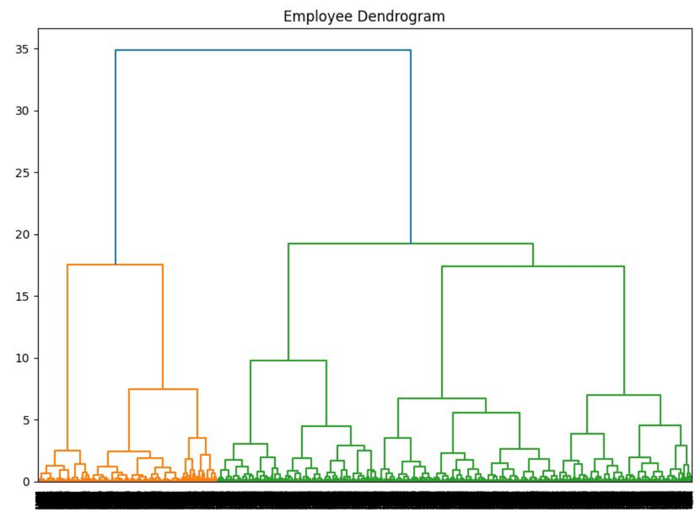
The K-Means algorithm starts with a specified number of centroids for future clusters, k, chosen randomly (but not too randomly). Data points are assigned to the centroid to which they are closest and this is repeated until there is no more movement.

1. Compute the distance between the centroids and all points in the dataset. Assign data points to the cluster whose centroid they are closest to.
2. Compute the new centroid of your clusters. This is done by taking the mean of all values for all tuples in that cluster. This is done so that the centroid represents all data points in the cluster.
3. All the distances between the new centroids and the data points is computed. Data points are assigned again to the centroid they are closest to, this may mean data points move from one cluster to another.
4. Steps 2 and 3 are repeated until there is no change in the membership of clusters (or a certain number of iterations has been reached to avoid huge computational costs).

Centroids are chosen in a way to keep the square-error Criterion at a minimum for each cluster formed. This is found using the formula to the right. It is the sum of the squared distances between all points and the centroid.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

Section 4A - Presenting results - Joseph

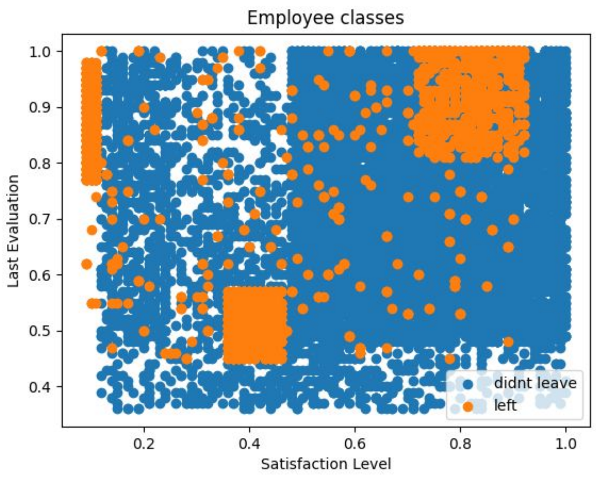


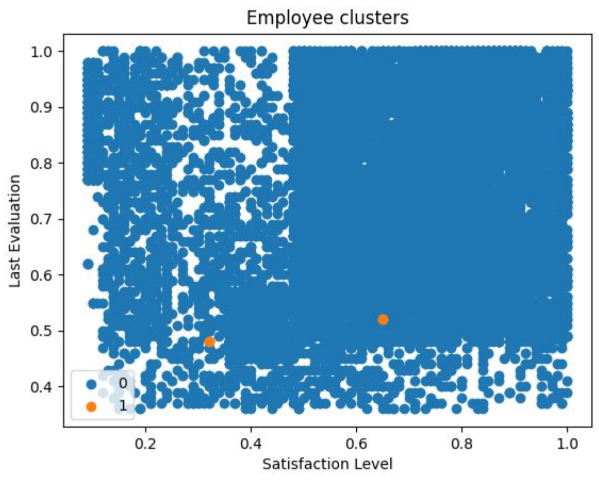
Although in this assignment we are aware of the number of classes prior to implementing clustering, I wanted to check that two clusters would be the optimal number using a dendrogram. I obtained the following:

The Dendrogram suggests that 2 clusters would be the most suitable number to extract. The dendrogram was produced on the satisfaction_level and last_evaluation attributes.

My plan initially was to use Gower distance to measure the distance between points but upon attempting this I obtained an interesting error: *"Memory error: 15.6 GiB of memory cannot be allocated to (14999, 14999, 9) nd array"*
The distance measure which I intended to use produced a tensor which was too large for my computer's RAM to process. As A result of this, I decided to attempt to use Manhattan distance instead and OneHotEncoding/Label Encoding for categorical variables where necessary. Based on our analysis of the dataset, I decided to drop the department variable at this point.

Results:
Correct clusters: 11424 / 14999
Accuracy: 0.761650



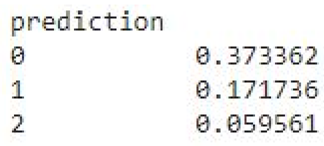


Analysis of the clusters revealed that only 4 employees formed the “left” cluster. I thought that this is probably due to the use of Manhattan distance in an already normalized dataset. Upon using Euclidean distance my results were identical.
Trying a series of other linkage methods other than ‘single-link’ such as ‘ward’ I obtained much different results which had higher ‘left’ counts but lower accuracy due to false positives. Intuitively, most employees are likely to not quit so assuming all employees will not quit yields high naive accuracy. This is similar to the results above. The normalized nature of the data may have come to be a disadvantage when implementing agglomerative hierarchical clustering as the majority of the data is very clustered together.

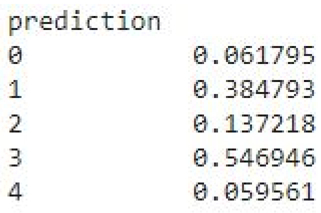
Section 4A - Presenting results - Adam

After implementing my algorithm I found that the dataset was not distributed very well and that upon having 2, 3 or even 4 clusters that the data was not grouped well according to whether an employee left or not. This I believe is as a result of the data not being a very friendly set and being a representation of a dataset we may find in the real world. It of course didn’t help that only 20% of all employees fell into the group that had left the company.

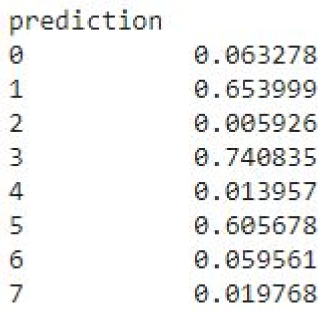
As a result, no cluster was labelled as a cluster of employees that left as less than half of the employees in all clusters belonged to the class “left”, as can be seen below to the left. When 5 clusters were used however I found that these clusters were more pure, and after 8 clusters it can be seen that a lot of clusters are almost pure. This shows to me that from all the different attributes there was no one combination that caused employees to leave. There were several pockets of employees that stayed and having 8 centroids allowed the algorithm to find them more accurately.



```
prediction
0          0.373362
1          0.171736
2          0.059561
```



```
prediction
0          0.061795
1          0.384793
2          0.137218
3          0.546946
4          0.059561
```



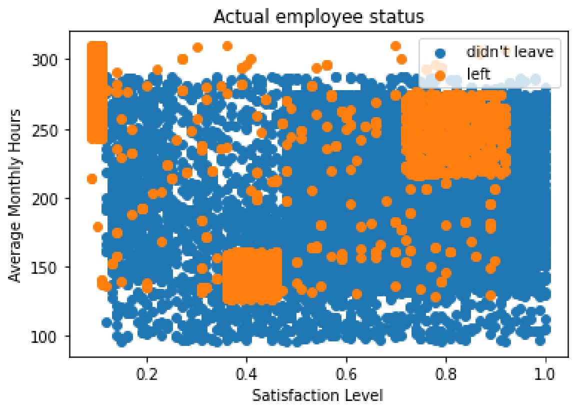
```
prediction
0          0.063278
1          0.653999
2          0.005926
3          0.740835
4          0.013957
5          0.605678
6          0.059561
7          0.019768
```

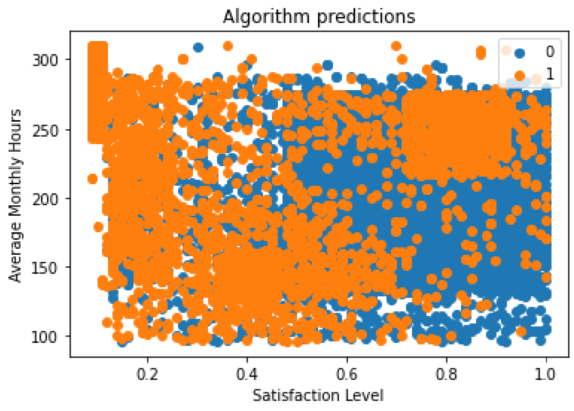
K-Means algorithm with k = 3, 5 and 8 respectively.

These different values for k produced accuracies from 76% at k=3 (Assuming all employees stayed), to 78% at k=5 and finally to 87% at k = 8. Typically clusters are taken between k=2 and k=6, although I am sure there are exceptions and I believe this may be such a time. Usually cluster accuracy and does not change much increasing k past around 6. In this case however, there is no 1 cluster of employees staying or going. So I believe having 8 clusters may increase its prediction accuracy.

Results:

```
Correctly predicted:13025 of 14999
Accuracy:0.8683912260817388
```





It can be seen from the actual employee status scatterplot that there were a couple of clear groupings of employees leaving. In the Algorithm predictions graph it can be seen that the square in the top right has been predicted, at the top left there is another correctly predicted set. Lastly, in the bottom middle you can see a high density of points corresponding to the square in the actual employee status graph. There are of course many points that were close to these that were labelled incorrectly, but this is always bound to be the case with clustering.

Comparison of the algorithms and Conclusion

It can be seen from the results of both algorithms that the dataset was difficult to work with. Both algorithms struggled to be accurate at low values of k and this is clear from our results. K-means clustering correctly identified a number of "pockets" of employees who are likely to leave but the accuracy suffered due to false positives. In contrast, Hierarchical clustering using single-link linkage clustered the vast majority of the dataset as "didn't leave" which suggests that the algorithm wasn't suited to the dataset.

The dataset itself has many different groupings of employees which are either leaving or staying. Which was of course an issue for a clustering algorithm with a low number of clusters as several of these groups are put into 1 cluster and the prediction becomes too broad.

This in a way is a lesson for future projects of ours. There can be many pockets of the one label sprinkled across a dataset, especially when working with a large number of variables. For example employees with lots of hours were likely to leave, but also employees with low hours who were not satisfied were likely to leave. So there is no work around this with a small number of clusters.

Additionally, the data itself is not a sure way of telling whether an employee will leave or stay. A person's personal circumstances cannot always be captured in a dataset. People with low hours, high satisfaction and performance and a high salary may still leave. So there will always be tuples that cannot be accurately predicted with any algorithm when the tuples represent people.