Clustering

Mark Priestley

INTRODUCTION

The dataset I am using contains information about loan applicants. I want to cluster similar Loan Applicants, with each cluster being classed as one of very likely, likely or unlikely to be accepted.

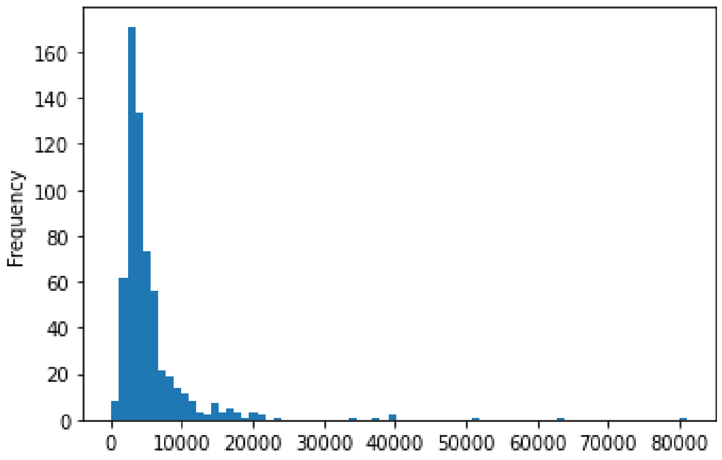
The dataset contains categorical/binary attributes ‘Married’, ‘Gender’, ‘Education’, ‘Self Employed’, ‘Property Area’ and ‘Loan Status’. It also contains interval-scaled attributes ‘Loan ID’, ‘Dependents’, ‘Applicant Income’, ‘Co Applicant Income’ and ‘Loan Amount’. I will use interval-scaled attributes ‘Applicant Income’, ‘Co Applicant Income’, ‘Loan Amount’ and Binary attributes ‘Education’ and ‘Credit History’ for my algorithms.

I plan on using K Means Clustering and Agglomerative Hierarchical Clustering to group the tuples based on how likely they are to be accepted for a loan. I will look at the percentage of tuples in each cluster that had their loan accepted, and see if the clusters have different probabilities of being accepted.

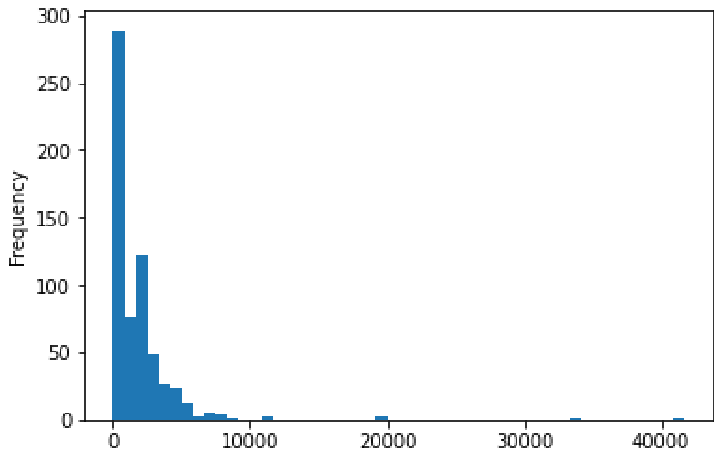
Analysis

Distribution of Interval Scales Variables

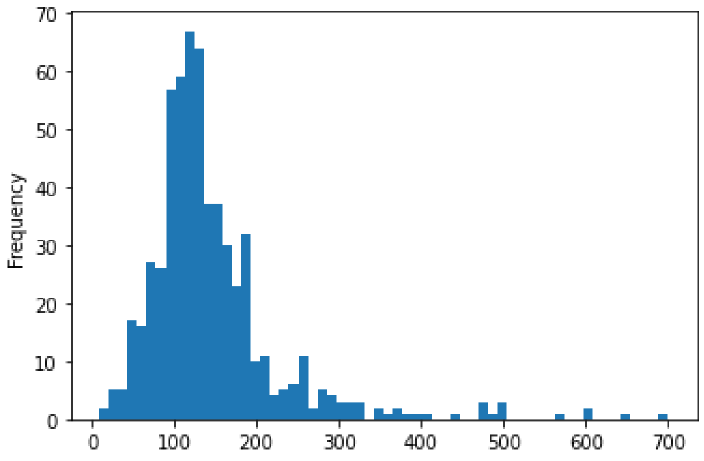
Applicant Income



Co Applicant Incom



Loan Amount



Because of the wide spread of the interval scaled variables, I had to standardize those attributes.

Standardization

Applicant Income, Co Applicant Income and Loan Amount have a very wide range of values, so they have to be standardized.

I did this by finding the mean absolute deviation of each of these attributes, and using this to find the z score.

	Applicant Income	Co Applicant Income	Loan Amount
Mean Absolute deviation	3034.90	1561.37	53.92
Standard Deviation	6109.04	2926.25	84.18
Mean	5403.46	1621.25	145.47

I used the mean absolute deviation opposed to the median absolute deviation, as I wanted the extreme outliers to remain detectable. For example, a value of 12841 in Applicant Income is 9806.1 above the mean. Using the mean absolute deviation gives a z score of 2.45. If I had used the standard deviation, it would have a z score of 1.22, which doesn't reflect how extreme the case is compared to the rest of the data in that attribute.

Distance Measurement Used - Interval Scaled Attributes

The two distance measurements I considered using for my algorithms were euclidean distance and manhattan.

After the data had been standardized, I used the euclidean distance to find the distance between the different interval scaled attributes. This is, for tuples x and y, calculated by $\sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$.

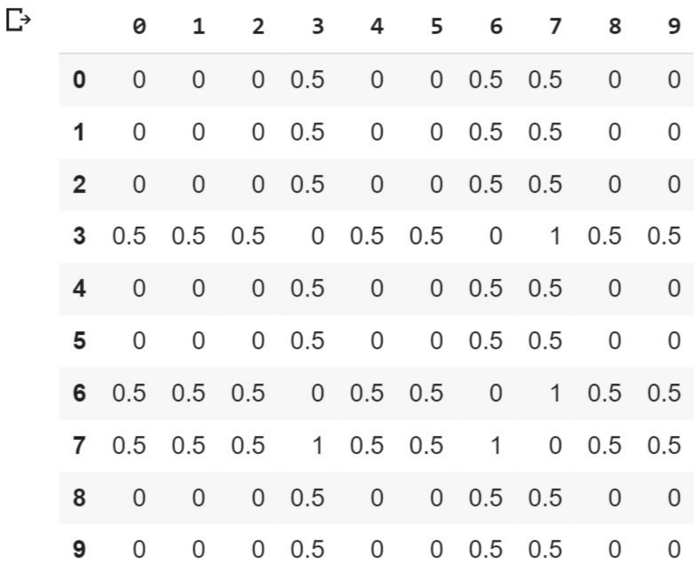
The distance between the different attributes are squared in euclidean distance, while with manhattan distance, the absolute values of the different attributes are summed. This reduces the effect of outliers between attributes, but since the data values were standardized, I choose to use euclidean distance.

Distance Measurement Used - Binary Attributes

The binary variables I chose to use in my dataset were 'Credit History', with possible values 1 and 0, and 'Education', with possible values 'Graduate' and 'Not Graduate'. I treated these as symmetric binary values.

One approach is to find the dissimilarity of different tuples. If two tuples, x and y, had one binary value the same and another different, would have a dissimilarity value of 0.5. If both values were the same, it would have a dissimilarity value of 0.

The dissimilarity of the values could be used to make a dissimilarity matrix. For example, the dissimilarity matrix of the first 10 values would look like,



	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0.5	0	0	0.5	0.5	0	0
1	0	0	0	0.5	0	0	0.5	0.5	0	0
2	0	0	0	0.5	0	0	0.5	0.5	0	0
3	0.5	0.5	0.5	0	0.5	0.5	0	1	0.5	0.5
4	0	0	0	0.5	0	0	0.5	0.5	0	0
5	0	0	0	0.5	0	0	0.5	0.5	0	0
6	0.5	0.5	0.5	0	0.5	0.5	0	1	0.5	0.5
7	0.5	0.5	0.5	1	0.5	0.5	1	0	0.5	0.5
8	0	0	0	0.5	0	0	0.5	0.5	0	0
9	0	0	0	0.5	0	0	0.5	0.5	0	0

Where the d[0, 0] equals 0 (tuples are the same) and d[i, j] is equal to d[j, i].

However, because the attributes I have selected are of mixed data types, I decided to treat the binary values as discrete values, and add greater weighting to ‘Credit History’. Using euclidean distance on binary values may give misleading clusters, so I will see at the end if the results are negatively impacted.

Algorithms

K means Clustering

One way of determining the ‘centre’ of a cluster is by defining its centroid. The centroid of a cluster is the point where each attribute is the mean value of that attribute of all the points in the cluster

K means clustering tries to minimize the distance between the centroid and all of the points in the cluster.

Firstly, I selected 3 tuples that were far apart, and made these tuples the centroids of the clusters. I then looped through each tuple in the dataset and found the euclidean distance between the tuple and each cluster. I assigned each tuple to it’s closest cluster.

I then repeated the following steps

- Find mean values for each attribute in each cluster to create a new tuple, and make that point the centroid of each cluster. This point might not be in the dataset.
- Loop through the dataset, assigning each tuple to it’s closest cluster.

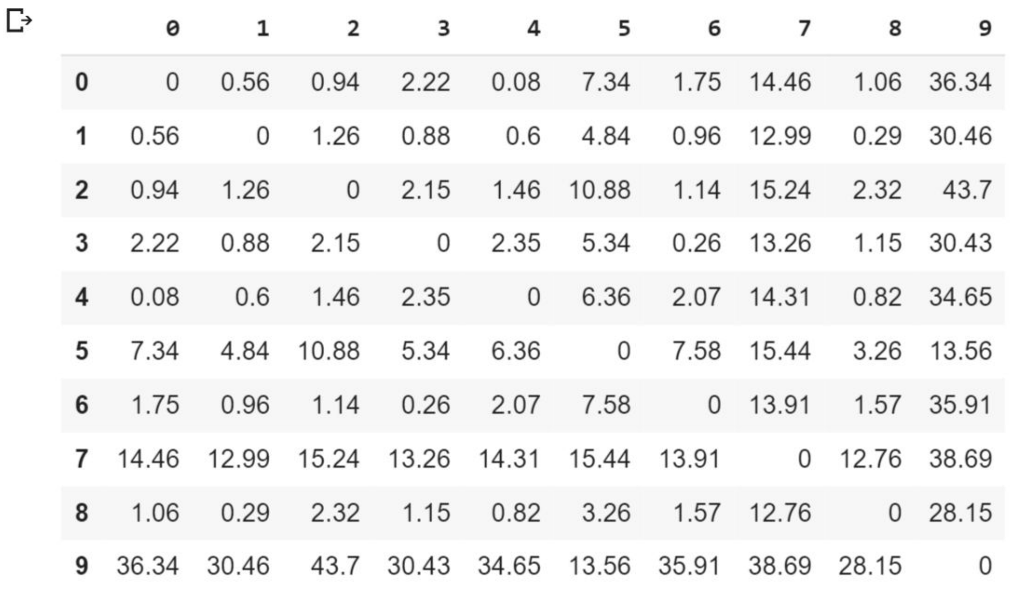
This stopped when the centroids remained the same before and after the iteration through the dataset.

Agglomerative Hierarchical Clustering

Each individual tuple starts off as its own cluster in Agglomerative Hierarchical Clustering. The distance between each tuple is put into a distance matrix.

The closest clusters are joined, and again, the distances between the clusters are put into a distance matrix. The distance between the clusters that were not joined in the last matrix remain the same, so there is no need to recalculate the distances for every cluster, only the newly formed cluster.

For example, the distance matrix for the first 10 tuples in the dataset would be,



	0	1	2	3	4	5	6	7	8	9
0	0	0.56	0.94	2.22	0.08	7.34	1.75	14.46	1.06	36.34
1	0.56	0	1.26	0.88	0.6	4.84	0.96	12.99	0.29	30.46
2	0.94	1.26	0	2.15	1.46	10.88	1.14	15.24	2.32	43.7
3	2.22	0.88	2.15	0	2.35	5.34	0.26	13.26	1.15	30.43
4	0.08	0.6	1.46	2.35	0	6.36	2.07	14.31	0.82	34.65
5	7.34	4.84	10.88	5.34	6.36	0	7.58	15.44	3.26	13.56
6	1.75	0.96	1.14	0.26	2.07	7.58	0	13.91	1.57	35.91
7	14.46	12.99	15.24	13.26	14.31	15.44	13.91	0	12.76	38.69
8	1.06	0.29	2.32	1.15	0.82	3.26	1.57	12.76	0	28.15
9	36.34	30.46	43.7	30.43	34.65	13.56	35.91	38.69	28.15	0

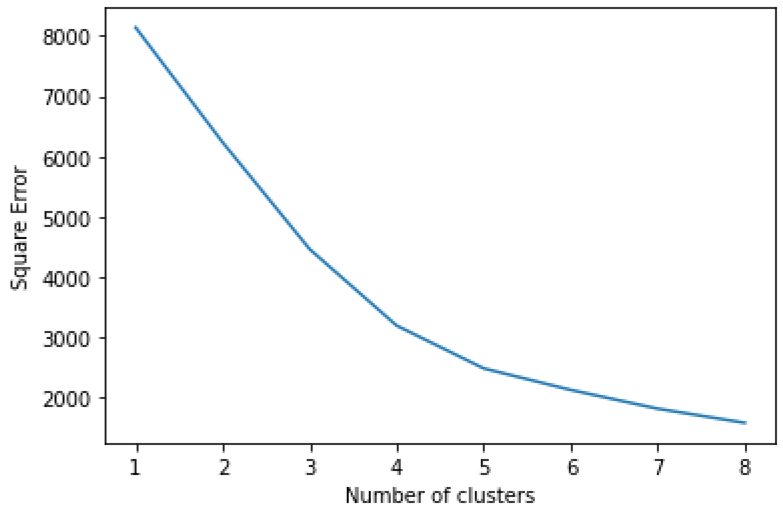
In this case, the algorithm would cluster d[0, 4]. However, this is only the first 10 tuples, the algorithm was incorporated on the entire dataset.

I used single-link clustering to find the distance between different clusters, which is the shortest distance between any member of one cluster to any member of the cluster.

Results

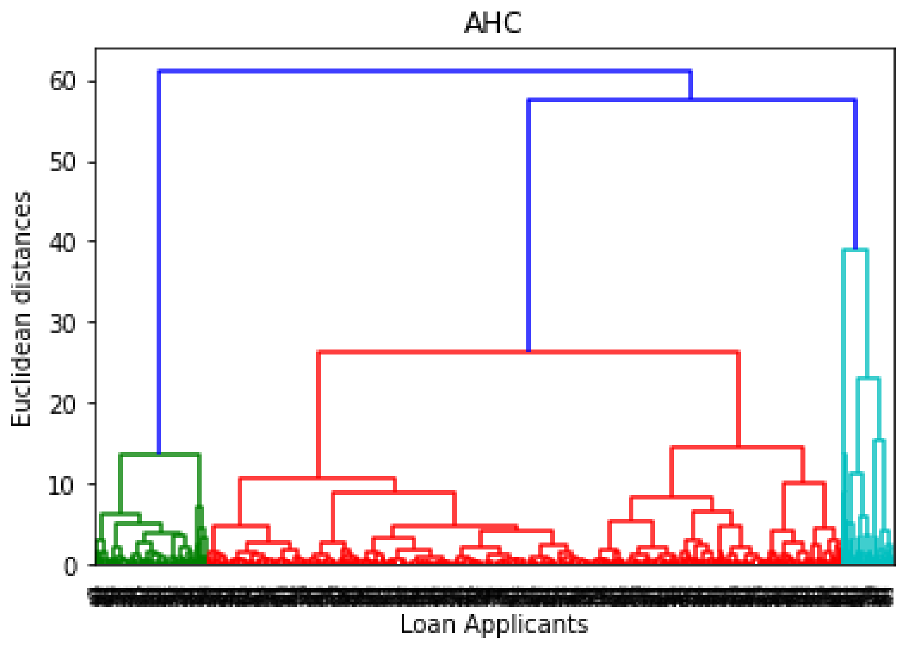
Accuracy of clusters

The sum of the square error for the 3 clusters of K Means Algorithm is 4432.88.



By increasing the number of clusters to 4 or 5, it would be below 3000, but I left the number of clusters at 3 in order to compare results with the AHC clusters.

Agglomerative Hierarchical Clustering Dendrogram



From the dendrogram, I think 3 was the correct number of clusters to use for AHC.

K Means clustering

The centroids produced by the K Means Clustering algorithm were as follows,

Cluster	Centroid	Tuples in Cluster
0	[0.7882352941176469, -0.229992156862745, 0.018572549019608198, -0.16459411764705903, 4.999999999999998]	510
1	[0.7011494252873562, -0.3243448275862069, -0.02779310344827591, -0.1417126436781609, -1.7763568394002505e-15]	87
2	[1.0, 8.559588235294115, -0.409529411764706, 5.663352941176471, 4.411764705882353]	17

The attributes were [Education, Applicant Income, Co Applicant Income, Loan Amount Credit History].

Cluster 0 is the largest cluster produced. It mainly contained Graduates, close to average values for Applicant Income, Co Applicant Income and a Credit History of 1. (I added a weighting for Credit History, so values of 1 equal 5)

Cluster 1 is also mainly Graduates, has a lower than average Applicant Income, and about average Co Applicant and Loan Amount. It has a lower Credit History value compared to the other clusters.

Cluster 2 is all Graduates, a higher than average Applicant Income, lower than average Co Applicant Income, and mainly a Credit History of 1.

How each cluster affects the likelihood of getting a loan

After grouping by clusters, I checked to see how the clusters produced by K Means affected ‘Loan Status’.

	Cluster	Loan_Status
0	0.0	0.792157
2	2.0	0.705882
1	1.0	0.068966

Across the entire dataset, 68.7% of loans were accepted.

Only 6.9% of tuples in cluster 1 had their loan application accepted, while70.6% of cluster 2 and 79.2% of cluster 0 were accepted.

Comparing the Clustering algorithms

I checked to see whether the 2 algorithms put the tuples in similar clusters. The list of the assigned clusters for the first 20 clusters were,

AHC clusters: [2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2]

K Means Clusters: [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

Here, cluster 2 of AHC resembles cluster 0 of K Means.

I found that 96.4% of the tuples of the AHC algorithm were assigned to the corresponding cluster in K Means.

Conclusion

The clusters formed may have formed more accurate clusters if I had incorporated the binary values and discrete valued differently rather than using the euclidean distance. Despite this, the clusters formed do have different chances of being accepted, with cluster 1 being unlikely, cluster 2 being likely and cluster 1 very likely to be accepted for a loan.