Identifying Poisonous Mushrooms using K-means Clustering

By: Ben, Filip, Daksh

Introduction and Goal:
Our group chose to implement the k-means clustering algorithm to cluster mushrooms by their attributes. We used a dataset containing over 60 thousand mushrooms, each with 20 attributes describing qualities such as stem height, gill color, habitat, etc. Each mushroom in the dataset also had a classification as either poisonous or edible. Our goal was to use k-means clustering to group mushrooms based on their 19 other (non-classification) attributes, with the hope that clusters would correlate strongly with the mushroom's classification as either poisonous or edible.

How the Algorithm works:
To accomplish the goal of identifying poisonous/edible mushrooms we implemented K-means clustering. The algorithm works by:

1. Identifying k points from the data set to serve as centroids of the clusters.
2. Iterating through each datapoint and comparing the distances to each centroid. Each datapoint is placed in the cluster with the closest centroid.
3. Centroids are updated by assigning them the mean of value for each attribute (for numeric attributes) or the mode of each value (for nominal attributes) of all the data points in that cluster.
4. Steps 2-3 are repeated until the stopping criteria is met. In this case, until there is no change in centroids, no change in datapoint assignment or the max number of iterations has been hit.

Implementing the algorithm with our data:
Our dataset includes 61,069 mushrooms, each with 20 variables (17 nominal, 3 metric). Due to the data being both nominal and metric we had to implement multiple distance methods. For the metric variables we used a basic Euclidean distance method. For the nominal variables we calculated the nominal distance (number of mismatching attributes / total number of attributes). Finally to calculate the total distance we normalized the nominal and euclidean distances and combined them. One caveat that we found was that we got the best results when not using all of the mushroom variables due to only some of them indicating if a mushroom is poisonous or not.

Due to K-means only working with quantitative data we had to edit the algorithm to work with all the Mushroom variables. To do this we implemented K-Prototypes which combines K-means (quantitative data) and K-modes (qualitative data). This means centroids are recomputed by taking the mean for numeric attributes and the mode for nominal attributes.

Code Breakdown:

Our code was separated into 3 classes:

1. **Mushroom Class** - Used to store data about a single mushroom. Includes 20 instance variables for all the mushroom attributes such as Cap Diameter and Cap Shape. The rest of the class consisted of distance methods and getters/setters.
2. **Cluster Class** - Used to store data about a single cluster. Includes an ArrayList of Mushroom objects belonging to the cluster and a single Mushroom object acting as the centroid.
3. **Mushroom Classification Class (Main Class)** - Contained the main method used to run the program. It also held other static methods to act on the data. This included the kmeans() method that implements the algorithm, readData() which reads the data from the csv, and the evaluateClusters method which outputs metrics such as the entropy and purity on the algorithm with different values of k.

We also used 3 different distance methods:

1. **Euclidean Distance** - Used to calculate the distance between quantitative attributes. Implemented as sqrt((x-y)^2).
2. **Nominal Distance** - Used to calculate the distance between nominal attributes. Implemented as (number of mismatching attributes / number of total attributes).
3. **Combined Distance** - Combination of euclidean and nominal distance used to calculate the total distance between Mushroom objects.

We also used various methods to evaluate the output of the algorithm:

1. **Density** - Used to calculate the density of the clusters. Implemented as sqrt(squareSum) / number of mushrooms in a cluster.
2. **Largeness** - Shows how many data points are in each cluster.
3. **Entropy** - Used to calculate the entropy for an ArrayList of clusters. Implemented by calculating the entropy for each cluster and then summing the individual cluster entropies multiplied by their respective weights to get the total entropy.
4. **Purity** - Used to calculate the purity for an ArrayList of clusters. Implemented by calculating the purity for each cluster and then summing the individual cluster purities multiplied by their respective weights to get the total purity.
5. **Ratio** - Shows the ratio between poisonous and not poisonous mushrooms in a cluster. This was our most important metric as it directly measured what we were optimizing for. In this case, we wanted clusters to be composed of mostly poisonous or mostly edible mushrooms.
6. **Silhouette Score -** The silhouette score metric measures how much the data overlaps. We calculated it with the formula (b-a) / max(b,a) for each point in the dataset where a is the distance between that point and every other point in its

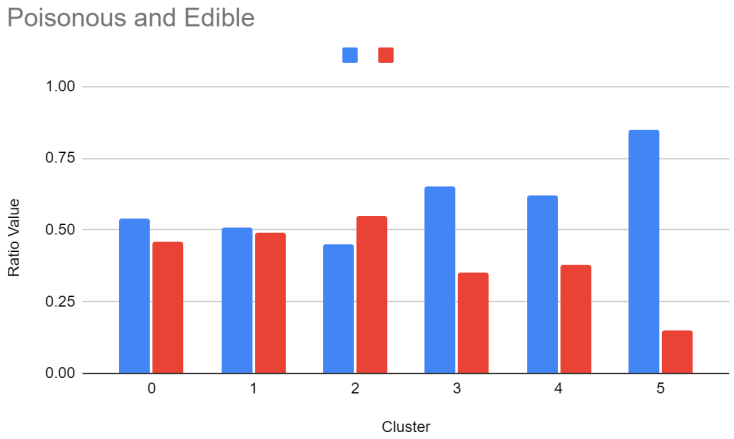
cluster and b is the distance between that point and every other cluster's centroid. A score of 1 means that the clusters are separated and distinct. 0 means they overlap, and -1 means that points have been wrongly assigned. We used this metric to determine the best k value, picking the lowest k with the highest silhouette score.

Major Obstacles:
We ran into a lot of difficulty trying to have the clusters separated by poisonous or edible mushrooms. Initially all our clusters had near even distribution of both poisonous and edible mushrooms. After some testing we found that we got better results when only using a smaller selection of Mushroom attributes. The subsection of attributes we used were determined by researching which qualities of mushrooms are the best indicators of their edibility.

The 9 attributes considered when obtaining our best algorithm output were all nominal. As a result a K-modes approach from the beginning would have sufficed without the need for Euclidean/Combined distance methods.

Results:

After evaluating the purity, entropy, average silhouette score, and density for clusters produced at different k-values, we determined k=6 produced the best clusters.
Analyzing the clusters produced by the algorithm with k=6, we found that around half of the clusters produced fairly accurate classifications of either poisonous or edible, but the other clusters still had relatively even distributions of both classes.



Conclusion:

Looking more closely at the dataset and the nature of mushrooms, we concluded that the k-means approach is not the best-suited algorithm for this problem. Poisonous mushrooms and edible mushrooms may share many of the same attributes, so classifying them based on the distance between their attributes does not always produce accurate results. K-means works best with data containing groups more clearly distinguishable by the differences in their attributes. Classifying mushrooms may work better with algorithms such as random forests or neural networks.