CPS 844 Lab 8

04/06/2022
Section 1
Ryan Soliven
500840952

#Part 1: Hierarchical Clustering methods (50 points)

# (0 point) Import the vertebrate.csv data

Code
```
data = pd.read_csv('vertebrate.csv')
```

# (5 points) Pre-process data: create a new variable and bind it with all the numerical attributes (i.e. all except the 'Name' and 'Class')

Code
```
NumericalAttributes = data.drop(['Name', 'Class'], axis=1)
```

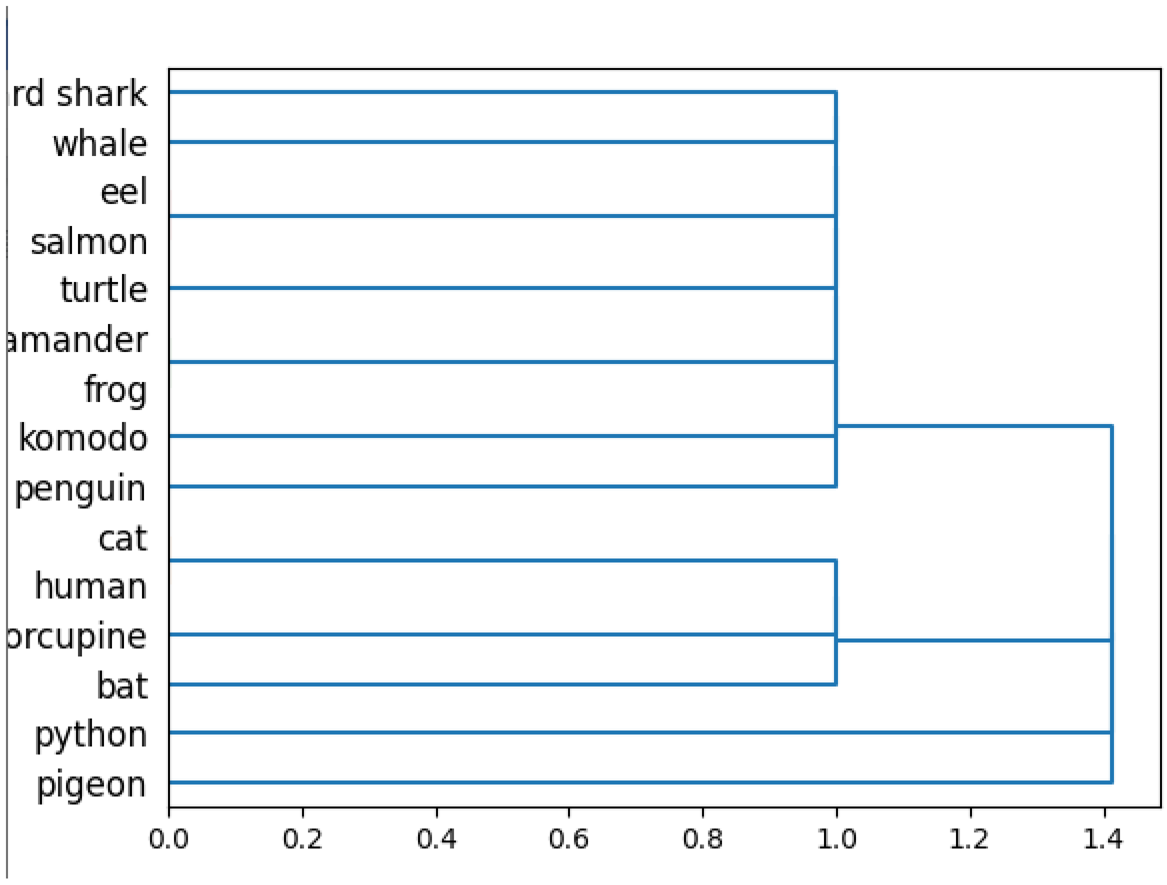
### (10 points) Single link (MIN) analysis + plot associated dendrogram ###

Code
```
min_analysis = hierarchy.single(NumericalAttributes)
```

# (5 points) Plot the associated dendrogram.
# Hint1: Make sure each data point is labeled properly (i.e. use argument: labels=data['Name'].tolist())
# Hint2: You can change the orientation of the dendrogram to easily read the labels: orientation='right'

Code
```
dn = hierarchy.dendrogram(min_analysis, labels = data['Name'].to_list(),
orientation='right')
plt.show()
```

Results



### (10 points) Complete Link (MAX) analysis + plot associated dendrogram ###

Code

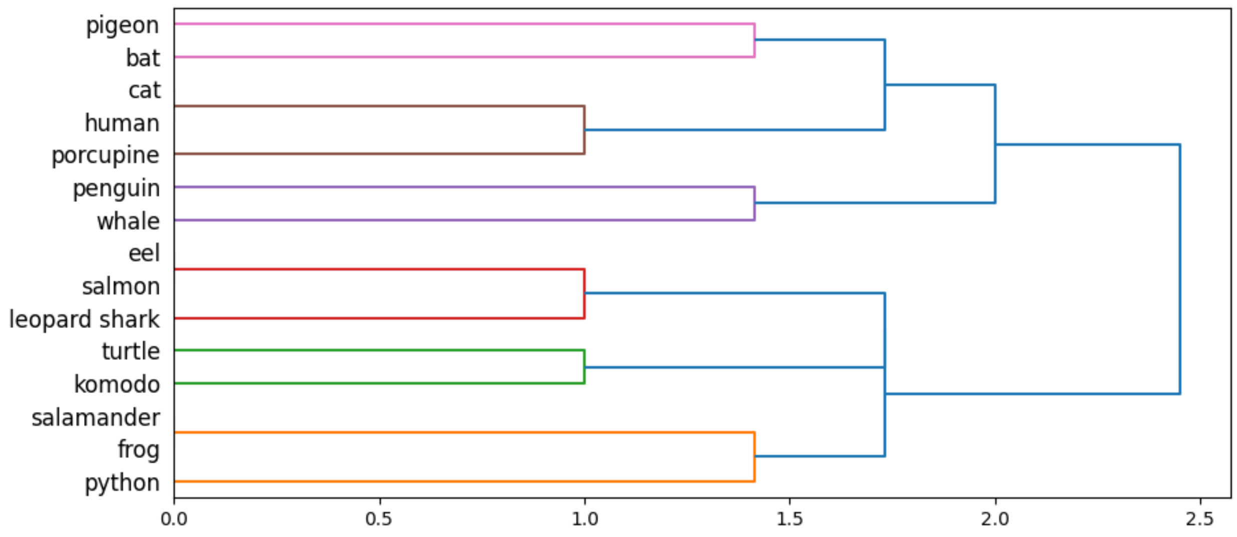
max_analysis = hierarchy.complete(NumericalAttributes)

# (5 points) Plot the associated dendrogram.
# Hint1: Make sure each data point is labeled properly (i.e. use argument: labels=data['Name'].tolist())
# Hint2: You can change the orientation of the dendrogram to easily read the labels: orientation='right'

Code

dn = hierarchy.dendrogram(max_analysis, labels = data['Name'].to_list(), orientation='right')
plt.show()

Results



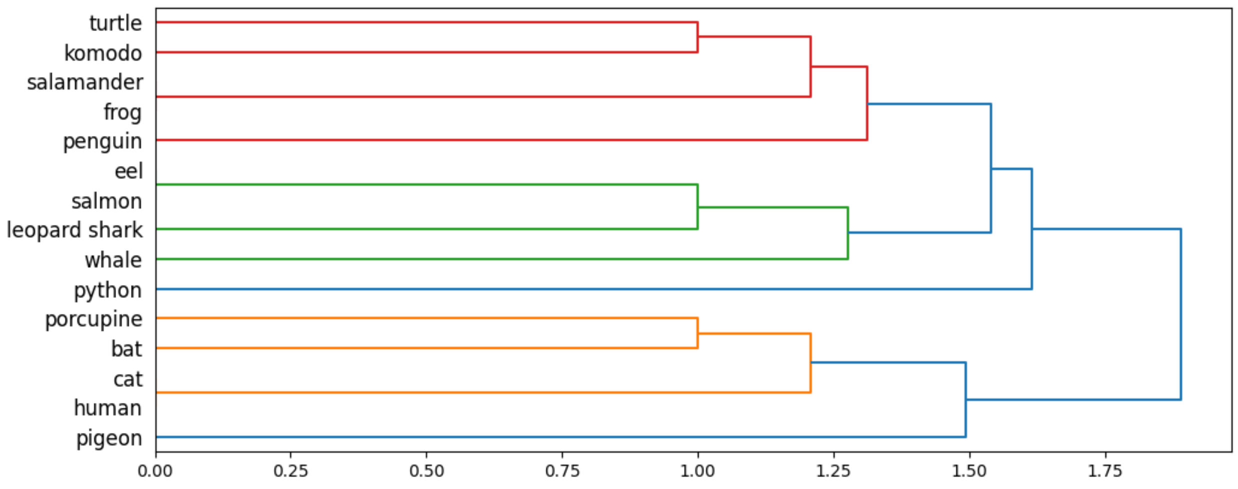
#### (10 points) Group Average analysis ###

Code
average_analysis = hierarchy.average(NumericalAttributes)

# (5 points) Plot the associated dendrogram.
# Hint1: Make sure each data point is labeled properly (i.e. use argument: labels=data['Name'].tolist())
# Hint2: You can change the orientation of the dendrogram to easily read the labels: orientation='right'

Code
```
dn = hierarchy.dendrogram(average_analysis, labels = data['Name'].to_list(),
orientation='right')
plt.show()
```

Results



# Part 2: Density-Based Clustering methods (50 points)

# (0 point) Import the chameleon.data data

Code

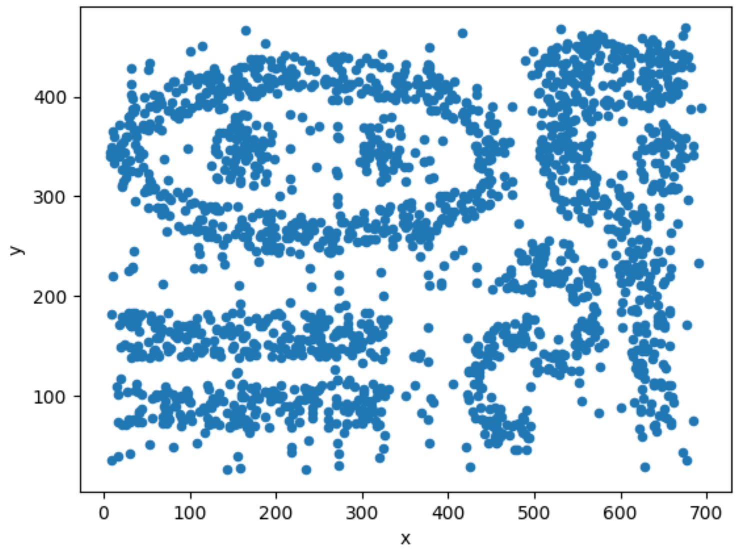
data = pd.read_csv('chameleon.data', delimiter=' ', names=['x','y'])

# Check the data distribution

data.plot.scatter(x='x',y='y')

plt.show()

Result



<u># (15 points) Apply DBScan: eps set to 15.5 and minpts set to 5.</u>
DBScanAnalysis = DBSCAN(eps=15.5, min_samples=5).fit(data)

<u># Concatenate data with cluster labels:</u>
<u># 1. Convert labels as a pandas dataframe</u>

<u>Code</u>
clustersLabels = pd.DataFrame(DBScanAnalysis.labels_,columns=['Cluster ID'])

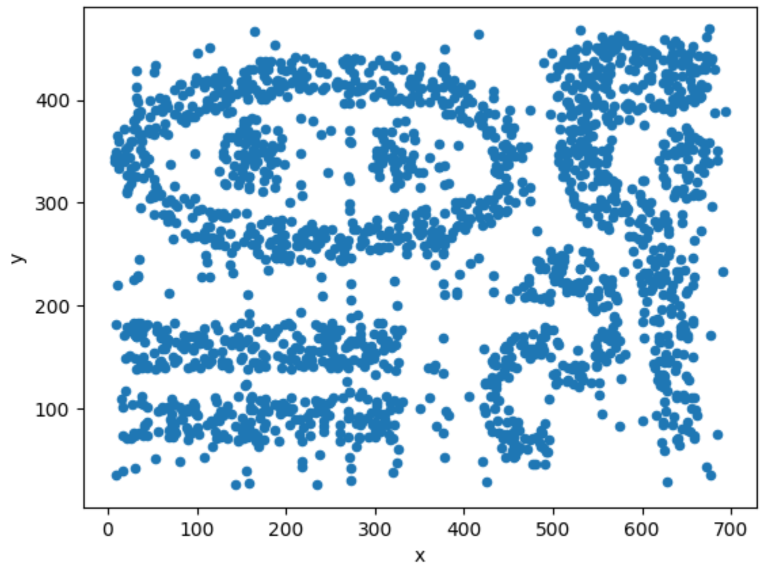
<u># 2. (15 points) Concatenate the dataframes 'data' and 'clustersLabels' (hint: use 'axis = 1' for concatenating along the column axis)</u>

<u>Code</u>
result = pd.concat((data, clustersLabels), axis=1)

<u># (10 points) Create a scatter plot of the data:</u>
<u># each point with coordinates x and y is represented as a dot;</u>
<u># use the value in 'Cluster ID' to color the point</u>
<u># Hint: the command is very similar to the one on line 17</u>

<u>Code</u>
result.plot.scatter(x='x',y='y', colormap='jet')
plt.show()

<u>Result</u>



# (10 points) How many clusters were found? Fill out the blank to tell, and don't include the noise points in the count.
# There are 9 clusters, not including the noise