E178/ME292B Final Project Report

Team 12: Austin Kim, Brady Johnson, Rushawn Childers, Yuqi Jin

Table of Contents

1 Introduction

This project aims to provide an implementation and analysis of the various clustering techniques for animal classification and to explore potentially how these methods can be improved using Gradient Boosting. In this project, we will perform a classification analysis of animal data using three different methods: k-means, Gaussian mixtures, and random forest. The data consists of various properties and features of each animal, which will be used to create models and group them into different classes.

K-means is a clustering algorithm that partitions data into k-clusters based on their similarity. Gaussian mixtures, on the other hand, use a probabilistic approach to clustering, where data points are modeled as a mixture of Gaussian distributions. Lastly, random forest is a classification algorithm that constructs a multitude of decision trees at training time and outputs the class which is the mode of the classes of the individual trees.

The accuracy of each method will be evaluated to determine which one performs best for this particular classification task. The results of the analysis will provide insight into the properties and features that differentiate the different animal classes and can be useful for various applications, and give us insight into the methods of classification and clustering.

2 Description of the Data

In this project, we used two data for this project: zoo.csv and class.csv. The dataset zoo.csv is a set of various animals, while the class.csv dataset contains the species types for all of the animals for the zoo.csv dataset. Both of these datasets derive from a Zoo Animal Classification dataset posted on *Kaggle*, a collaboration platform that has a large variety of datasets with many different applications.

2.1 zoo.csv Dataset

The zoo.csv dataset is organized in alphabetical order and comprises 101 animals, with each animal characterized by 16 animal-type features expressed in numerical form. The “leg” feature indicates the number of legs an animal has, while the “catsize” feature details if the animal is larger than a cat, with a value of 1 indicating yes and a value of - indicating no. The “class_type” feature, which ranges from 1 to 7, represents the animal's specific classification. Additional information regarding class types can be seen in the class.csv section. The remaining features are binary, with a value of 1 indicating the presence of the feature and 0 indicating its absence. The visualization of the dataset can be seen in the figure below.

	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	class_type
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1

2.2 class.csv Dataset

This data provides a description of the 'class_type' feature found in the 'zoo.csv' dataset. It contains information on the correspondence between class numbers and their corresponding animal classifications, the number of animals in each class, and a list of the animals belonging to each class. The value “Class_Number” is representative of the “class_type” value from the zoo.csv dataset. A partial view of the data is provided in the figure below.

Class_Number	Number_Of_Animal_Species_In_Class	Class_Type	Animal_Names
1	41	Mammal	aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hai
2	20	Bird	chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, spar
3	5	Reptile	pitviper, seasnake, slowworm, tortoise, tuatara
4	13	Fish	bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna
5	4	Amphibian	frog, frog, newt, toad
6	8	Bug	flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp
7	10	Invertebrate	clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm

3 Methodology

3.1 K-Means Clustering

K-means is usually implemented to start with a random centroid for n-defined clusters. Check the Euclidean distances from each data point mainly on a 2-dimensional plane (2 columns for the x and y-axis). The next centroid is chosen from the data points with a probability proportional to the square of the distance to the nearest centroid. Data points that are farther away from the existing centroids will then be chosen as the next centroid. This is repeated until there are n clusters as defined in the beginning.

For our data set, we have more than 2 features and we use 16 animal features to classify an animal. There are 7 classifications as shown in the class.csv. Given the high dimensionality of the problem going beyond the 3 dimensions, we knew we would not be able to visualize the clustering results in the usual and colorful manner. To do these classifications we used scikit-learn's k-means clustering model. Though it may have not been robust enough to handle our dataset given that this classification method resulted in the lowest accuracy score.

3.2 Gaussian Mixtures

We implemented Gaussian Mixtures Methods with and without scikit-learn library. For GMM without scikit-learn, we assume that the distribution of class-conditioned weights is Gaussian to use this method. Given that each animal has sixteen features, $y_i \in \Re^{16}$ and the covariant matrix has 16 by 16 dimensions. In order to implement the Expectation-Maximization algorithm, we initialize the clusters $\{(\mu_k, \Sigma_k, \pi_k)\}_K$ as follows: $\pi_k = 1/7, \Sigma_k = I$ (identity matrix), and μ_k as the mean of animal features, classified in the K-means method to make better results even though we could use random values. This initialization is performed in the 'initialize_clusters' function. Also,

to calculate the $N_k(y_i)$, we defined a function named 'gaussian,' which calculates normal distribution pdf with this formula: $N_k(y) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(y - \mu)^T \Sigma (y - \mu))$.

To repeat the expectation step and a maximization step, we defined two functions for each step. First, in a function "expectation_step". we calculated the γ_{ik} for each k with this formula: $\gamma_{ik} = (\pi_k N_k(y_i))/(\sum_{j=1}^{K} \pi_j N_j(y_i))$ and stored it in the global variable 'gamma_ik.' Also, we assigned the value of γ_{ik} nominator in the 'totals' variable to calculate likelihood value later.

Second, for the maximization step, we defined 'maximization_step' function and calculated N_k, π_k, μ_k, and Σ_k. The variables calculated as follows; $N_k = \sum_{i=1}^{N} \gamma_{ik}$, $\pi_k = \frac{N_k}{N}$, $\mu_k = \frac{1}{N_k}\sum_{i=1}^{N} \gamma_{ik} y_i$, $\Sigma_k = \frac{1}{N_k}\sum_{i=1}^{N} \gamma_{ik}(y - \mu_k)^2$, which gained from partial derivatives of the equation $\sum_{i=1}^{N} ln\left(\sum_{k=1}^{K} \pi_k N_k(y_i)\right) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)$, to maximize it with respect to λ, μ_N, π_k, and Σ_N. These values are updated in the variable 'clusters.'

Finally, we defined 'train_gmm' to implement the EM algorithm, and we set $K = 7$ because this project aims to classify animals into seven given classes. For each epoch, we calculated the value of likelihood with the formula of $L = \sum_{i=1}^{N} ln\left(\sum_{k=1}^{K} \pi_k N_k(y_i)\right)$. The results are shown in the next section.

3.3 Random Forests

The algorithm is based on creating a "forest" of decision trees, where each tree is trained on a random subset of the data and a random subset of the features. The final classification is then based on the majority vote of all the trees in the forest.

First, start by collecting data from the zoo.csv dataset. Then clean the data by removing any irrelevant values. In this case, we created a pure_data_2 variable by removing animal names and class types. The next step is to split the data into training and prediction sets. In this experiment, we do three different ways to split the data: training and prediction with a full data set, with 80% and 20 % and 60% and 40%, respectively.

After that, we started doing the model evaluation for each one to evaluate the model's performance on the testing data by calculating metrics' accuracy. Then, by doing the tree visualization to make random forests more comprehensive. Train a random forest model on the training data. The model will create a forest of decision trees, each trained on a random subset of the data and features.

4 Results and Analysis

4.1 K-Means Clustering Findings

We were able to successfully predict each animal's classification from the zoo.csv with no issues or errors. Since this is a high-dimensional dataset with 16 features for each characteristic, we were not able to visualize the results on a two or even three-dimensional plot. However, we were able to show the clusters in a text as seen in Table 1. However, when examining the clusters, you can very clearly see animals that do not belong in the same group. For instance, in Cluster 6 - where there is a crab, frog, newt, toad, tortoise, and tuatara - if you look at the class.csv you can see that frogs, newts, and toads are amphibians. But a crab is an invertebrate, while a tortoise and tuatara are reptiles. Just from that, we can see how inaccurate the predictions from K-means really are. When calculating the accuracy score, it can range from 0.6 to 0.01, with the state randomized, often falling below 0.33 out of say 13 runs.

Cluster 0:	Cluster 1:	Cluster 2:	Cluster 3:	Cluster 4:	Cluster 5:	Cluster 6:
chicken	aardvark	dolphin	crayfish	clam	bass	crab
crow	antelope	porpoise	flea	pitviper	carp	frog
dove	bear	seal	gnat	seasnake	catfish	frog
duck	boar	sealion	honeybee	seawasp	chub	newt

flamingo	buffalo		housefly	slowworm	dogfish	toad
fruitbat	calf		ladybird	slug	haddock	tortoise
gull	cavy		lobster	worm	herring	tuatara
hawk	cheetah		moth		pike	
kiwi	deer		octopus		piranha	
lark	elephant		scorpion		seahorse	
ostrich	giraffe		starfish		sole	
parakeet	girl		termite		stingray	
penguin	goat		wasp		tuna	
pheasant	gorilla					
rhea	hamster					
skimmer	hare					
skua	leopard					
sparrow	lion					
swan	lynx					
vampire	mink					
vulture	mole					
wren	mongoose					
	opossum					
	oryx					
	platypus					
	polecat					
	pony					
	puma					
	pussycat					
	raccoon					
	reindeer					
	squirrel					
	vole					
	wallaby					
	wolf					

Table 1: Results from K-Means fitting the whole dataset.

4.2 Gaussian Mixtures Findings

Unfortunately, we were unable to proceed with the EM algorithm beyond the second epoch due to an overflow error. It appears that one of the clusters was focused on a single dataset, causing its covariance to converge to zero. We confirmed this suspicion by examining the covariance matrix and observing that the determinant values became extremely small, or some matrices were no longer valid *(figure 2)*. The discernibly low accuracy score of the Gaussian Mixture Method implemented using the scikit-learn library, as shown in above, further supports this observation *(figure 3)*.

```
n_clusters = 7
n_epochs = 3

clusters, likelihoods, scores, sample_likelihoods, history = train_gmm(np_data, n_clusters, n_epochs)
```

```
Epoch:  1 Likelihood:  -9617.138237522728
Epoch:  2 Likelihood:  -6661.716147463573
Epoch:  3 Likelihood:  nan
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_30388\553839977.py:10: RuntimeWarning: overflow encountered in exp
  return np.diagonal(np.exp(coeff + ex_power)).reshape(-1, 1)
C:\Users\USER\AppData\Local\Temp\ipykernel_30388\553839977.py:9: RuntimeWarning: invalid value encountered in scalar power
  coeff = np.log(1 / ((2 * np.pi) ** (n / 2) * np.linalg.det(cov) ** 0.5))
```

Figure 1: GMM could not be implemented

```
= epoch:  0
1.0
1.0
1.0
1.0
1.0
1.0
1.0

= epoch:  1
4.5255816987845045e-28
1.0241879514620574e-29
3.120094654400319e-24
7.299928051838917e-33
1.795569952519159e-25
7.398208998789419e-30
5.577017635254171e-22

= epoch:  2
0.0
0.0
-8.673296427269487e-135
0.0
3.222865947836424e-139
3.3316147688348275e-227
0.0
```

```
scores = np.vstack((scores, [score, 'RF_60']))
scores
```

```
array([['0.3465346534653465', 'K_F'],
       ['0.0', 'K_80'],
       ['0.37623762376237624', 'GM_F'],
       ['0.23809523809523808', 'GM_80'],
       ['0.37623762376237624', 'BGM_F'],
       ['0.23809523809523808', 'BGM_80'],
       ['0.23809523809523808', 'GM_E'],
       ['1.0', 'RF_F'],
       ['0.8095238095238095', 'RF_80'],
       ['0.8048780487804879', 'RF_60']], dtype='<U32')
```

Figure 2 (left): determinant of covariance for each epoch

Figure 3 (right): accuracy score for different methods

4.3 Random Forests Findings

The accuracy of splitting data is essential to building and evaluating modes shown in Figure 4. After the observation, we found that the accuracy of training and prediction with the full data set is higher than the 80/20 and 60/40 split. The accuracies are 1.0, 0.809, and 0.904 respectively. The higher accuracy can generalize well to new data. It is significant to provide insights into how well the model fits the data.

Training and Prediction with full data set

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(max_depth=None
                             ,random_state=rng_seed
                            )
clf.fit(Pure_data_2, Target)
clf.predict(Pure_data_2)
```

```
array([1, 1, 4, 1, 1, 1, 1, 4, 4, 1, 1, 2, 4, 7, 7, 7, 2, 1, 4, 1, 2, 2,
       1, 2, 6, 5, 5, 1, 1, 1, 6, 1, 1, 2, 4, 1, 1, 2, 4, 6, 6, 2, 6, 2,
       1, 1, 7, 1, 1, 1, 1, 6, 5, 7, 1, 1, 2, 2, 2, 2, 4, 4, 3, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 7, 4, 1, 1, 3, 7, 2, 2, 3, 7, 4, 2, 1, 7, 4, 2,
       6, 5, 3, 3, 4, 1, 1, 2, 1, 6, 1, 7, 2])
```

```
score = metrics.accuracy_score(Target,clf.predict(Pure_data_2))
score
```

```
1.0
```

Training and Prediction with 80/20 split

```
clf = RandomForestClassifier(max_depth=None
                             ,random_state=rng_seed
                            )
clf.fit(Pure_data_2.iloc[:80,:], Target.iloc[:80])
clf.predict(Pure_data_2.iloc[80:,:])
```

```
array([3, 6, 4, 2, 1, 7, 4, 2, 6, 5, 2, 5, 4, 1, 1, 2, 1, 6, 1, 6, 2])
```

```
score = metrics.accuracy_score(Target.iloc[80:],clf.predict(Pure_data_2.iloc[80:,:]))
score
```

```
0.8095238095238095
```

Training and Prediction with 60/40 split

```
clf = RandomForestClassifier(max_depth=None
                             ,random_state=rng_seed
                            )
clf.fit(Pure_data_2.iloc[:60,:], Target.iloc[:60])
clf.predict(Pure_data_2.iloc[60:,:])
```

```
array([4, 4, 5, 1, 1, 1, 1, 1, 1, 1, 1, 2, 6, 4, 1, 1, 4, 7, 2, 2, 5, 6,
       4, 2, 1, 7, 4, 2, 6, 5, 2, 5, 4, 1, 1, 2, 1, 6, 1, 6, 2])
```

```
score = metrics.accuracy_score(Target.iloc[60:],clf.predict(Pure_data_2.iloc[60:,:]))
score
```

```
0.8048780487804879
```

Figure 4: The accuracies for model's performance on the testing data

In this experiment, we visualize our first three trees, shown in Figure 5. The max depth is chosen as 16 because there are 16 animal-type features in the dataset. Each node contains the following information: physical characteristics of animals and values used for splitting, the percentage of total samples in each split, and the percentage split between classes in each split. The root node contains information on animal features like legs, fins, hair, etc. The branches connecting the nodes represent the decisions made based on the values of the features. Then, for example, if the root node corresponds to

“legs<= 3.0”, the internal node represents decisions based on other characteristics, such as “hair<=0.5”. Moreover, the leaf nodes represent specific animal classes, such as “mammals,” “reptiles,” “birds,” etc. When the nodes’ color gets darker, the node gets closer to being fully 0 or 1.

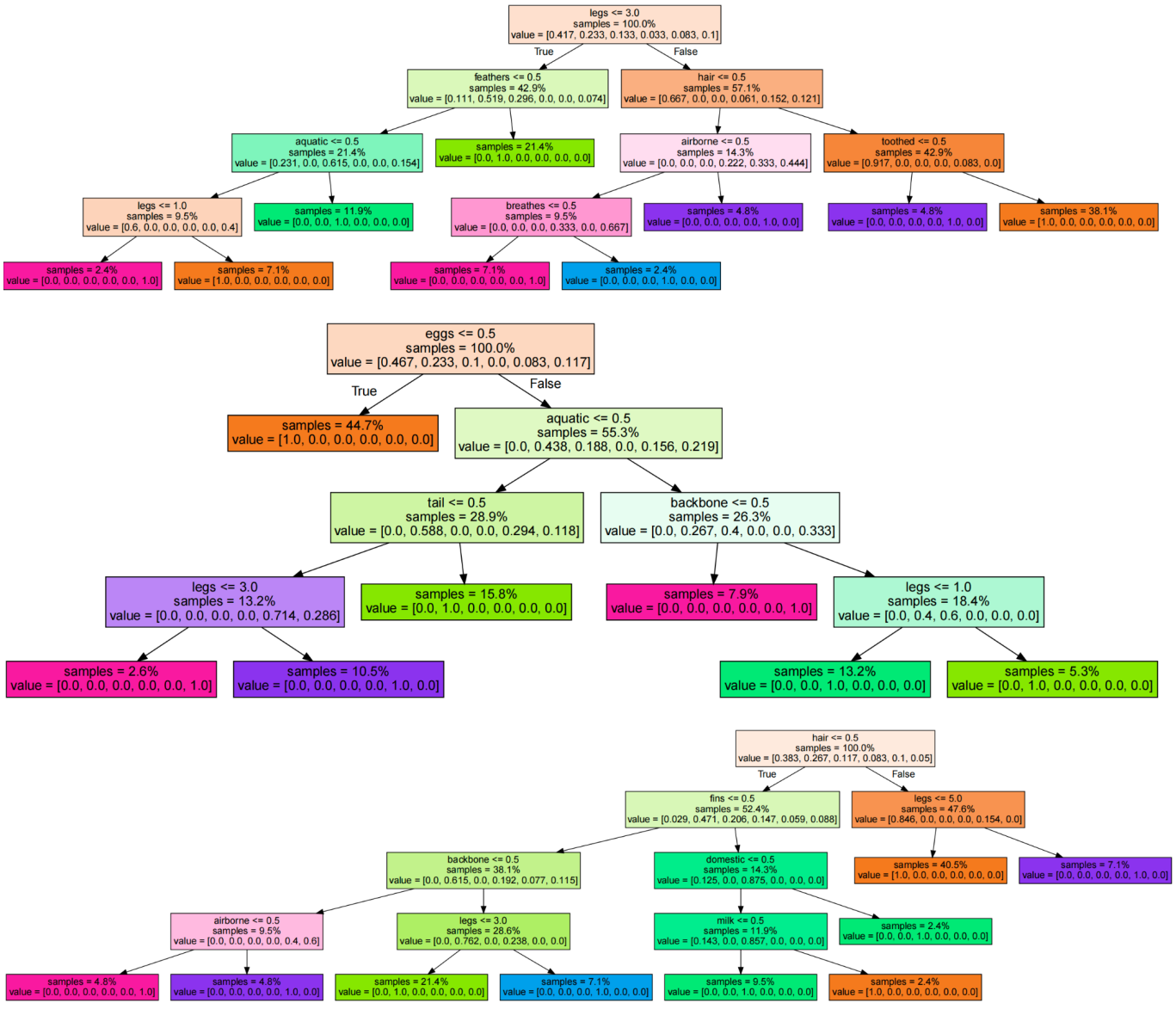

Figure 5: Random Forest Visualization of Zoo Classification

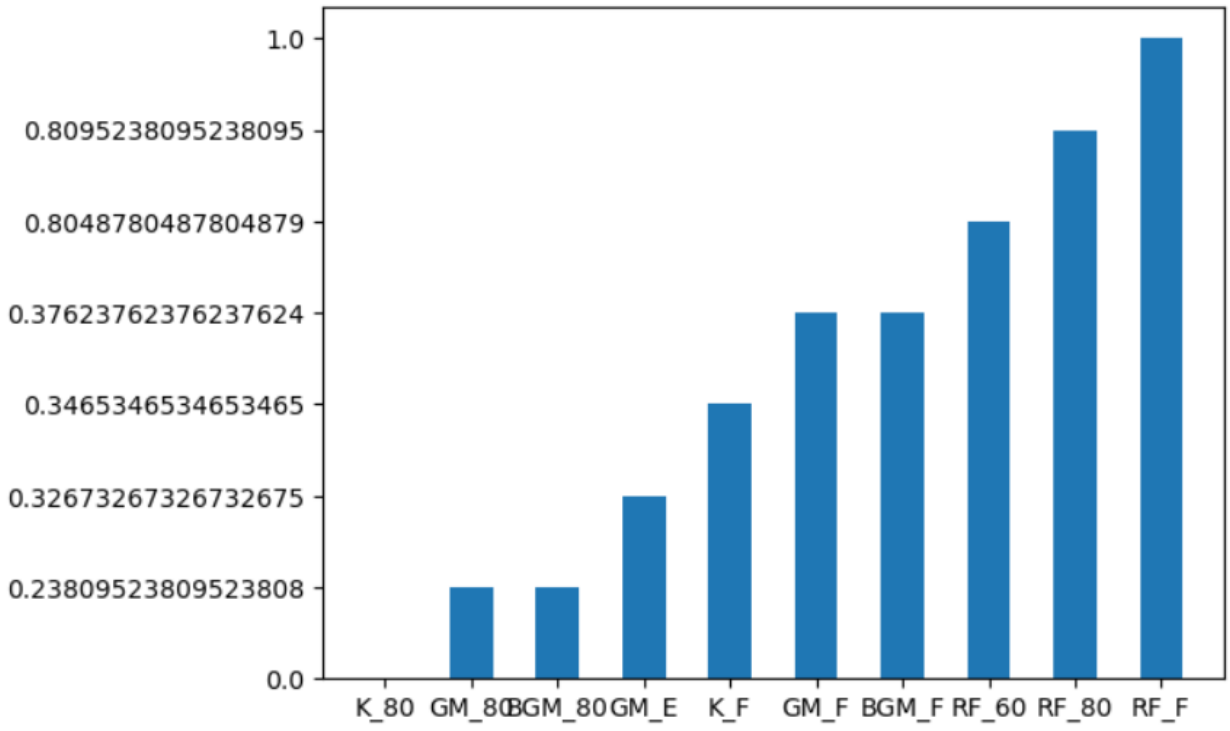

Figure 6: Bar chart of all methods with different test train splits.

Figure 6 is a bar chart showing all the accuracy scores of each model we tested on the zoo.csv. We used 3 main classifiers K-means, Gradient Mixture (GM), Random Forest (RF), and additionally Bayesian Gaussian Mixture (BGM). Bayesian Gaussian Mixture was added last minute to see if there was an accuracy difference between it and Gradient Mixture, which there does not seem to be. For all the models, we stuck with a random_state = 454 for consistency. We were very surprised that K-means underperformed so greatly. Especially when we did an 80/20 train test split into the data, which resulted in a zero accuracy score for some reason. In the same vein, we were blown away by the impressive accuracy scores from the RF model. In hindsight, since our data mainly consists of binary and discrete numbers it makes sense that a decision tree would have been a better first choice in a prediction model rather than K-means which we originally picked.

5 Areas for Future Research

If our group had more time to participate in this project, we would have diversified the type of datasets we use for our three means of clustering. Our first choice of clustering was using K-means, and we found that it was not the most effective means to cluster our data. This, however, may not hold true with other kinds of datasets. By performing these operations on a large variety of datasets, we can further realize our findings and perhaps yield a different observation than we currently have. By having more datasets, we could have also found more similar datasets to the ones we used to use even more types of animals.

6 Conclusion

The objective of this report was to develop a better understanding of clustering algorithms and to compare different methods of clustering. Concluding this project, we believe we have successfully reached this objective. Clustering algorithms remain pertinent to data exploration and information retrieval within datasets which can have a wide application within engineering, marketing, and other economic sectors. Our group has gained a lot from the efforts in this project and hope to carry them forward within the future of our engineering endeavors.

7 References

[1] "Zoo Animal Classification." *Www.kaggle.com*, www.kaggle.com/datasets/uciml/zoo-animal-classification.

[2] ocontreras309. "ML Notebooks." *GitHub*, 14 Mar. 2023, github.com/ocontreras309/ML_Notebooks/blob/master/GMM_Implementation.ipynb. Accessed 30 Apr. 2023.