*Random Forest: A Powerful Machine Learning Algorithm*

*Olivia Brague, Stephora Alberi, Jade Pearl*

**ABSTRACT**

The intention of this group project was to engage in research of random forest, a supervised machine learning algorithm. This was the group’s first encounter with algorithms of the machine learning family, so the territory of researching such an algorithm was entirely unfamiliar and took some time but it did not remain daunting as more research had been gathered. In order to help further our understanding of the functions and overall implementation of random forest, the group had gathered as much research as possible by looking for articles, example code, and videos that could help shed light on the random forest algorithm. Then we implemented our algorithm in Python and discussed it together. This paper discusses how we went about researching the algorithm and what we now understand about it.

**INTRODUCTION**

Machine learning is an up-and-coming field of computer science that many are still becoming exposed to for the first time; including us, since this was our first in-depth encounter with machine learning. While those who study computer science may be familiar with older data structures and algorithms, many machine learning and artificial intelligence algorithms are novel within the computer science industry, an ever-growing subject and field of study. Even today, plenty of research is being carried out and attempts at understanding these algorithms are being made. Although research regarding machine learning is readily available and at our disposal, it may not be accessible or as easy to understand for less experienced computer scientists that have not had as much experience in this field. It is important to create material and research that is understandable for most programmers so that they too can utilize these algorithms and gain a higher level of understanding of the boundaries of machine learning and artificial intelligence.

Coming to an understanding of machine learning algorithms such as random forest can be difficult as available research does not always use understandable language, i.e., the diction is too practical for those of lower experience to comprehend easily. Programmers also learn in diverse ways and what may be approachable for one person may not be the same for another. The first instinct may be to simply look it up and hope for the best, but that usually does not lead to comprehension of the algorithm if you are a beginner in the field. Something very helpful to counteract the difficulty in understanding is to find articles and videos that simplify the concepts of random forest.

Other contributors to research regarding the algorithm would have done well to keep in mind that their readers may not always be as learned or as knowledgeable as they are. It is within their better interest to use comprehensible language to not alienate readers. It would be desirable for them to act more as a guide to future computer scientists studying in that field. While many papers have failed to follow such criteria, we seek to cover this gap with our experience and research. Our goal is to highlight our findings regarding the random forest algorithm in a concise and unambiguous manner.

Random forest is an incredibly useful algorithm, and we wish to help others reach the same level of understanding we have gained in our research. As dictated before, the drafting of this paper is equipped with needed information which presents our findings on random forest in as simple of a manner as possible whilst still fulfilling the duty of educating the user efficiently on this growing subject of importance. We hope that in doing this, others will be able to understand this algorithm as well as be able to use it, as many computer and data scientists have been making use of the power of random forest presently.

**SUMMARY OF CONTRIBUTIONS**

* Related Work
* Running Example
* Preliminaries
* Our Research Process
* The Implementation
* Performance Experiments
* Conclusion

**RELATED WORK**

Those who are interested in learning more about random forest should investigate sources such as the following:

<https://towardsdatascience.com/random-forest-in-python-24d0893d51c0>

<https://www.mygreatlearning.com/blog/random-forest-algorithm/#:~:text=A%20random%20forest%20classifier%20works,cannot%20be%20defined%20by%20classes>

<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20forest%20algorithm%20is%20an,both%20classification%20and%20regression%20problems>.

<https://machinelearningmastery.com/random-forest-ensemble-in-python/>

**RUNNING EXAMPLE**

We chose to use random forest to address a statistical problem that involves both classification and regression. Random forest is known for being powerful when used to put labels onto data (E R, 2021). The specific application of the algorithm we chose to use involves putting labels of class (mammal, bird, reptile, etc.) onto animals based on said animal's characteristics. We decided to find a dataset comprised of many animals so that we could work with the data and categorize each animal.

**PRELIMINARIES**

Random forest is what is known as a supervised machine learning algorithm (E R, 2022). This means that random forest must take in data provided by the user. The algorithm makes use of multiple decision trees, a type of greedy algorithm that picks between different options in the data in a tree-like format (Bento, 2022). It uses classification and regression, where classification takes in variables that can only end up being a couple of things, and regression takes in variables that could have any type of value (E R, 2022).

Another important aspect of implementing random forest is bagging and bootstrapping. Bagging and bootstrapping help with the variance experienced with decision trees, and as stated by Bento, 2022, “Each model is trained on a *different* dataset, because they’re bootstrapped. So inevitably, each model will make *different* mistakes, and have a distinct error and variance. Both the error and variance get reduced in the aggregation step where, literally in the case of Regression, they are *averaged* out.”

**OUR RESEARCH PROCESS**

When we first took on this project, our priority was in understanding the basics of random forest and the common functions and uses for it. We found primarily that most of the time, the algorithm is used by data scientists (E R, 2021). The specific uses from there are diverse, with many possibilities for what the implementation of random forest can be used for.

We thought it would be helpful to examine specific examples of how others have used random forest to solve problems. E R, 2021 describes how it can be used for data regarding commerce, banking, medicine, or the stock market, but those who wish to implement random forest are not limited to just these uses. E R, 2021 also focused on random forest being used for medical predictions, describing how the algorithm can help medical professionals discover how likely someone may be to suffer from heart disease.

Now that our understanding of the uses of the algorithm was out of the way, we wanted to learn how it was actually implemented. We came up with an idea for how we would use random forest and then we began to experiment with an online software known as Jupyter in order to write and test our code.

**THE IMPLEMENTATION**

The best way to come to an understanding of these algorithms is through the use of real-world examples. In implementing random forest, we produced our own example to explore: the classification of zoo animals. Online there are a few datasets belonging to zoos that hold information regarding their animals. From there, it is simple to see the categories those animals are sorted into and allow random forest to learn from those categories.

We imported libraries such as pandas into the notebook, as these libraries enable us to use their tools in order to more easily and efficiently create the algorithm. From there, we also imported our zoo dataset to manipulate and interpret the data within it. We believe that the classification of animals is an example that anyone could understand so that those who view the implementation can see random forest in action.

**PERFORMANCE EXPERIMENTS**

For our code, we used two datasets. The first dataset contained a list of 100 animals with their corresponding physical characteristics and class type number. The second dataset contained the class type numbers and their corresponding classification names. The class names are based on 7 general animal classifications which are mammals, fish, bugs, amphibians, reptiles, bird, and invertebrate. The first step in our code, we read in both of our datasets and assigned them to variables using the panda utility function, pd.read. We printed out both of datasets using their assigned variables to see the readability and organization of the data, and it was not easy to follow at first. So, we organized both of our datasets into data frames. Although, having to keep simultaneously looking back and front between datasets can get confusing. So, in addition, we laid all our classification information out and used that information to create a new column for our data frame containing the animal names and named it “species”. This column carries the class names of the corresponding animal based on its class type number. The next step we took was to split our data into training and testing sets. We gave twenty-five percent of our data to the testing dataset and the rest of the seventy-five percent to the training dataset. Once we did that, we were finally able to create our random forest. By default, a random forest can make up to one hundred decision trees. But for the purposes of our code, we stuck with 10. We then created a predict variable, named “y\_predict”, which will store the results of our random forest predictions. And based on that, we compare both our predicted results with our test data results to check our accuracy. Our accuracy score came out as one hundred percent, which is within the typical scale. As a plus, we printed out a confusion matrix based on our predicted and test data results to see a more extended explanation of our accuracy score. Usually, a typical confusion matrix will resemble a consistent matrix, meaning the diagonal will contain nonzero integers while the rest will contain only zeros. The diagonals represent the number of times the predicted and test data agreed on the same results. Any nonzero integer outside that diagonal represents the number of times the predicted and test data shared the same characteristics (inputs) but generated different results (outputs). This means that somewhere in the original dataset is inconsistent. Finally, at the end of our code, we print out all ten of our decision trees. At first glance, each of the decision trees is built completely differently, but the way that they’re able to generate predictions is the same. As a way of checking the accuracy of the decision trees manually, we choose any animal of our liking, whether it was in our dataset or not, and tested it in our decision tree. As a result, each animal we tested had the correct classification. However, there were a few animals, them being platypuses, spiders, and scorpions, that we had to exempt from testing due to their complex classifications. This can always be fixed by adding more data (in our original dataset) relative to these complex classifications.

**CONCLUSION**

We feel that in studying this algorithm, we have learned a lot about both random forest and machine learning in general. We also believe that the code we created is understandable in a way that some code written for the algorithm may not be. This can help others to also gain an understanding of what random forest is as well. It is a truly powerful algorithm with many uses and should be used more in the future.

**FUTURE WORK**

In writing our code, we wish that we could have got around to exploring how the algorithm works with other datasets. We only used the singular zoo dataset, but we feel that we could have looked at other zoos as well. This would make sure that the algorithm we created is accurate when different lists of animals are applied to it other than the one that we originally used. We also wish to have a way of entering an animal into the program in the future and having the algorithm indicate what category in the animal kingdom it thinks it belongs to.

**ACKNOWLEDGEMENTS**

Olivia contributed by doing research and constructing most of the research paper as well as the paper’s structure. Stephora contributed by doing research and writing a part of the paper as well as the code implementation and articulating the project demonstration for the presentation. Finally, Jade contributed by also doing research as well as putting together the PowerPoint and helping to guide the group along with paper edits.

All group members participated with each other as a whole in giving feedback and needed help to one another on the paper, PowerPoint, and code implementation. We have researched and practiced the presentation together on a constant basis leading up to the presentation and we were in constant discussion with one another while working on our individual tasks. The other group members had to readily be available for feedback, input, and/or editing whenever there was a significant change to one of the above pieces of the project. Communication was frequent, clear, and the group is glad to report that we are proud of our work on this project and understand a lot more of what machine learning is and the purpose of random forest.

**CITATIONS**

Bento, C. (2022, January 10). *Random forests algorithm explained with a real-life example and some python code*. Medium. Retrieved May 5, 2023, from <https://towardsdatascience.com/random-forests-algorithm-explained-with-a-real-life-example-and-some-python-code-affbfa5a942c>

E R, S. (2021, June 17). *Random Forest | Introduction to Random Forest Algorithm*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

**APPENDIX**

*#import scikit's Random Forest library*  
*from* sklearn.ensemble **import** RandomForestClassifier  
  
#Loading pandas  
import pandas **as** pd  
  
#loading numpy  
import numpy **as** np  
  
#loading tree visualizations  
import matplotlib.pyplot **as** plt  
  
#Loads model performance  
from sklearn.metrics **import** accuracy\_score, confusion\_matrix, ConfusionMatrixDisplay, classification\_report

In [121]:

*#Creating an object for zoo dataset*  
*zoo* **=** pd**.**read\_csv('zoo.csv')  
  
#Creating an object for zoo class dataset  
zoo\_class **=** pd**.**read\_csv('class.csv')  
  
#Printng out the dataset info  
print(zoo)  
print(zoo\_class)

animal\_name hair feathers eggs milk airborne aquatic predator \  
0 aardvark 1 0 0 1 0 0 1   
1 antelope 1 0 0 1 0 0 0   
2 bass 0 0 1 0 0 1 1   
3 bear 1 0 0 1 0 0 1   
4 boar 1 0 0 1 0 0 1   
.. ... ... ... ... ... ... ... ...   
96 wallaby 1 0 0 1 0 0 0   
97 wasp 1 0 1 0 1 0 0   
98 wolf 1 0 0 1 0 0 1   
99 worm 0 0 1 0 0 0 0   
100 wren 0 1 1 0 1 0 0   
  
 toothed backbone breathes venomous fins legs tail domestic \  
0 1 1 1 0 0 4 0 0   
1 1 1 1 0 0 4 1 0   
2 1 1 0 0 1 0 1 0   
3 1 1 1 0 0 4 0 0   
4 1 1 1 0 0 4 1 0   
.. ... ... ... ... ... ... ... ...   
96 1 1 1 0 0 2 1 0   
97 0 0 1 1 0 6 0 0   
98 1 1 1 0 0 4 1 0   
99 0 0 1 0 0 0 0 0   
100 0 1 1 0 0 2 1 0   
  
 catsize class\_type   
0 1 1   
1 1 1   
2 0 4   
3 1 1   
4 1 1   
.. ... ...   
96 1 1   
97 0 6   
98 1 1   
99 0 7   
100 0 2   
  
[101 rows x 18 columns]  
 Class\_Number Number\_Of\_Animal\_Species\_In\_Class Class\_Type \  
0 1 41 Mammal   
1 2 20 Bird   
2 3 5 Reptile   
3 4 13 Fish   
4 5 4 Amphibian   
5 6 8 Bug   
6 7 10 Invertebrate   
  
 Animal\_Names   
0 aardvark, antelope, bear, boar, buffalo, calf,...   
1 chicken, crow, dove, duck, flamingo, gull, haw...   
2 pitviper, seasnake, slowworm, tortoise, tuatara   
3 bass, carp, catfish, chub, dogfish, haddock, h...   
4 frog, frog, newt, toad   
5 flea, gnat, honeybee, housefly, ladybird, moth...   
6 clam, crab, crayfish, lobster, octopus, scorpi...

In [122]:

*#Viewing the top 10 rows of the dataset*  
*zoo***.**head(10)

Out[122]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 |
| **5** | buffalo | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 1 |
| **6** | calf | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 1 | 1 | 1 |
| **7** | carp | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 4 |
| **8** | catfish | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 4 |
| **9** | cavy | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 1 | 0 | 1 |

In [123]:

*#Viewing the*   
*zoo\_class***.**head(10)

Out[123]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Class\_Number** | **Number\_Of\_Animal\_Species\_In\_Class** | **Class\_Type** | **Animal\_Names** |
| **0** | 1 | 41 | Mammal | aardvark, antelope, bear, boar, buffalo, calf,... |
| **1** | 2 | 20 | Bird | chicken, crow, dove, duck, flamingo, gull, haw... |
| **2** | 3 | 5 | Reptile | pitviper, seasnake, slowworm, tortoise, tuatara |
| **3** | 4 | 13 | Fish | bass, carp, catfish, chub, dogfish, haddock, h... |
| **4** | 5 | 4 | Amphibian | frog, frog, newt, toad |
| **5** | 6 | 8 | Bug | flea, gnat, honeybee, housefly, ladybird, moth... |
| **6** | 7 | 10 | Invertebrate | clam, crab, crayfish, lobster, octopus, scorpi... |

In [124]:

print(zoo['class\_type'])

0 1  
1 1  
2 4  
3 1  
4 1  
 ..  
96 1  
97 6  
98 1  
99 7  
100 2  
Name: class\_type, Length: 101, dtype: int64

In [125]:

zoo\_class["Class\_Number"]**.**unique()

Out[125]:

array([1, 2, 3, 4, 5, 6, 7], dtype=int64)

In [126]:

zoo\_class["Class\_Type"]**.**unique()

Out[126]:

array(['Mammal', 'Bird', 'Reptile', 'Fish', 'Amphibian', 'Bug',  
 'Invertebrate'], dtype=object)

In [127]:

zoo['species'] **=** pd**.**Categorical**.**from\_codes(zoo**.**class\_type, categories**=** ['None','Mammal', 'Bird', 'Reptile', 'Fish', 'Amphibian', 'Bug',  
 'Invertebrate'])  
  
zoo**.**head(10)

Out[127]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **animal\_name** | **hair** | **feathers** | **eggs** | **milk** | **airborne** | **aquatic** | **predator** | **toothed** | **backbone** | **breathes** | **venomous** | **fins** | **legs** | **tail** | **domestic** | **catsize** | **class\_type** | **species** |
| **0** | aardvark | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 0 | 1 | 1 | Mammal |
| **1** | antelope | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 1 | Mammal |
| **2** | bass | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 4 | Fish |
| **3** | bear | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 0 | 1 | 1 | Mammal |
| **4** | boar | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 1 | Mammal |
| **5** | buffalo | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 1 | Mammal |
| **6** | calf | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 1 | 1 | 1 | Mammal |
| **7** | carp | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 4 | Fish |
| **8** | catfish | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 4 | Fish |
| **9** | cavy | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 1 | 0 | 1 | Mammal |

In [128]:

*#loading train and testing utility functions*  
*from* sklearn.model\_selection **import** train\_test\_split  
  
x **=** zoo**.**iloc[:, 1:17]  
  
y **=** zoo**.**iloc[:, 17]  
#Splitting dataset into a trainig dataset and test dataset  
x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size **=** 0.25, random\_state **=** 0)

In [129]:

rf **=** RandomForestClassifier(n\_estimators **=** 10, random\_state **=** 0)  
rf**.**fit(x\_train, y\_train)  
  
rf**.**estimators\_

Out[129]:

[DecisionTreeClassifier(max\_features='sqrt', random\_state=209652396),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=398764591),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=924231285),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=1478610112),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=441365315),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=1537364731),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=192771779),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=1491434855),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=1819583497),  
 DecisionTreeClassifier(max\_features='sqrt', random\_state=530702035)]

In [130]:

y\_predict **=** rf**.**predict(x\_test)

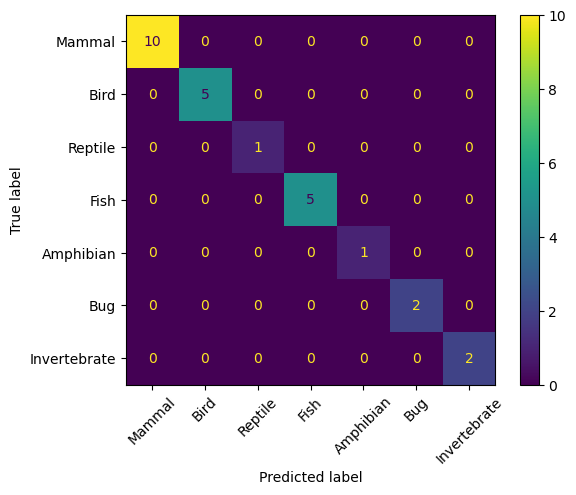
In [131]:

accuracy **=** accuracy\_score(y\_test, y\_predict)  
  
print("The accuracy of test data: ", accuracy)

The accuracy of test data: 1.0

In [132]:

cm **=** ConfusionMatrixDisplay(confusion\_matrix **=** confusion\_matrix(y\_test, y\_predict), display\_labels **=** zoo\_class["Class\_Type"]**.**unique())  
  
cm**.**plot(xticks\_rotation **=** 45)  
  
plt**.**show()



In [133]:

*#Creating a list of the feature column names*  
*features* **=** zoo**.**columns[1:17]  
  
#Viewing features  
features

Out[133]:

Index(['hair', 'feathers', 'eggs', 'milk', 'airborne', 'aquatic', 'predator',  
 'toothed', 'backbone', 'breathes', 'venomous', 'fins', 'legs', 'tail',  
 'domestic', 'catsize'],  
 dtype='object')

In [134]:

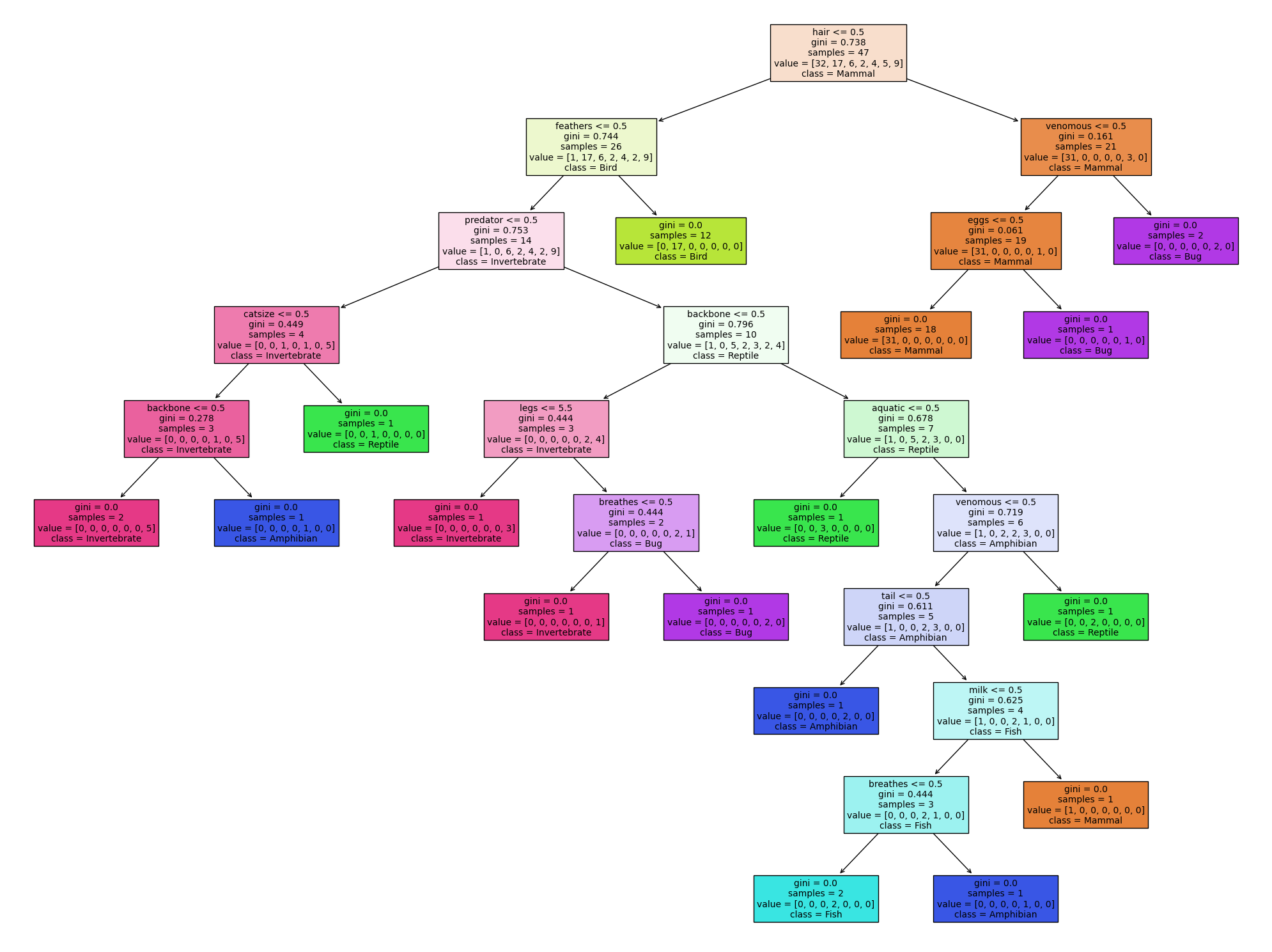
target **=** zoo\_class["Class\_Type"]**.**unique()  
  
target

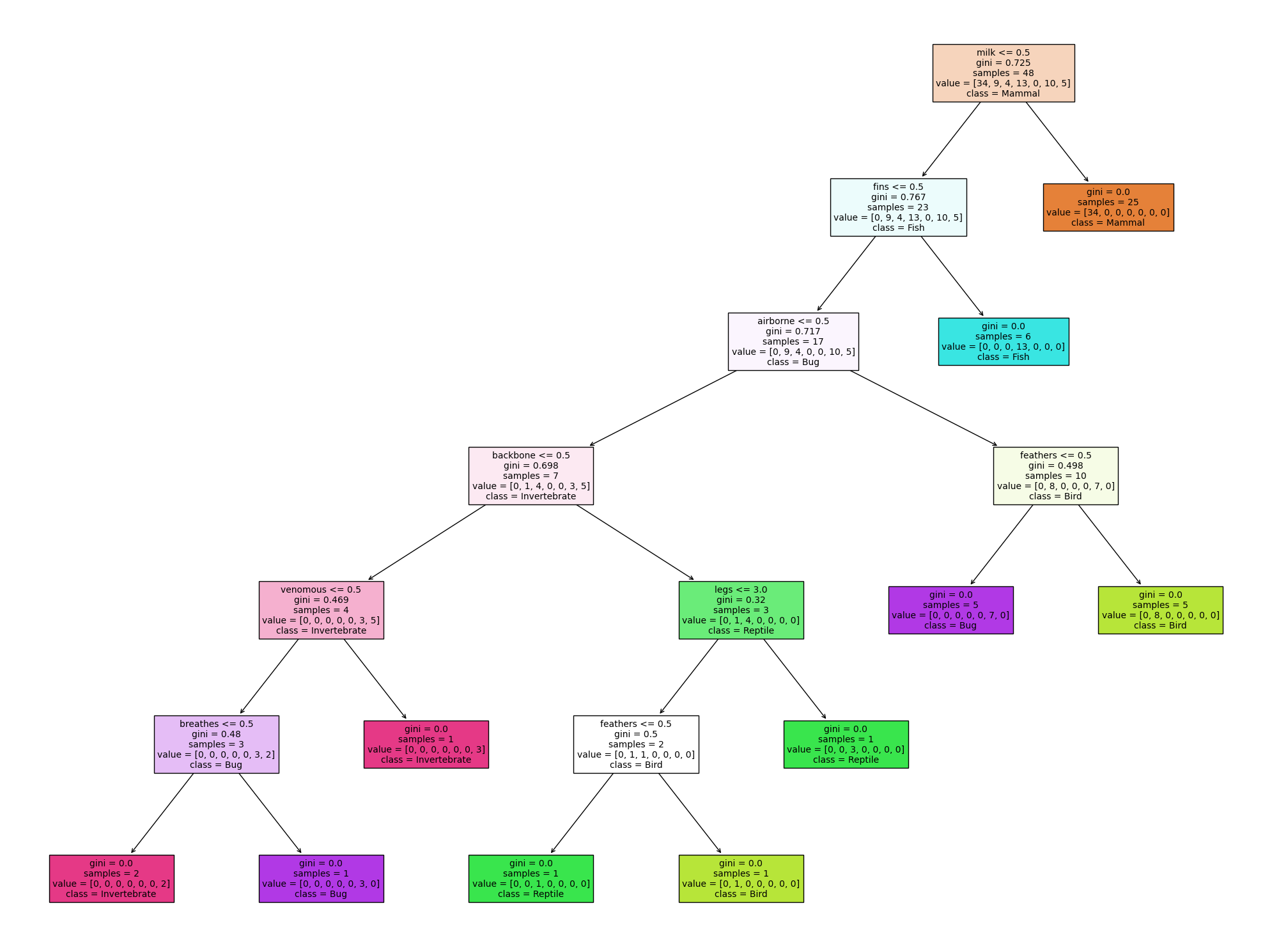
Out[134]:

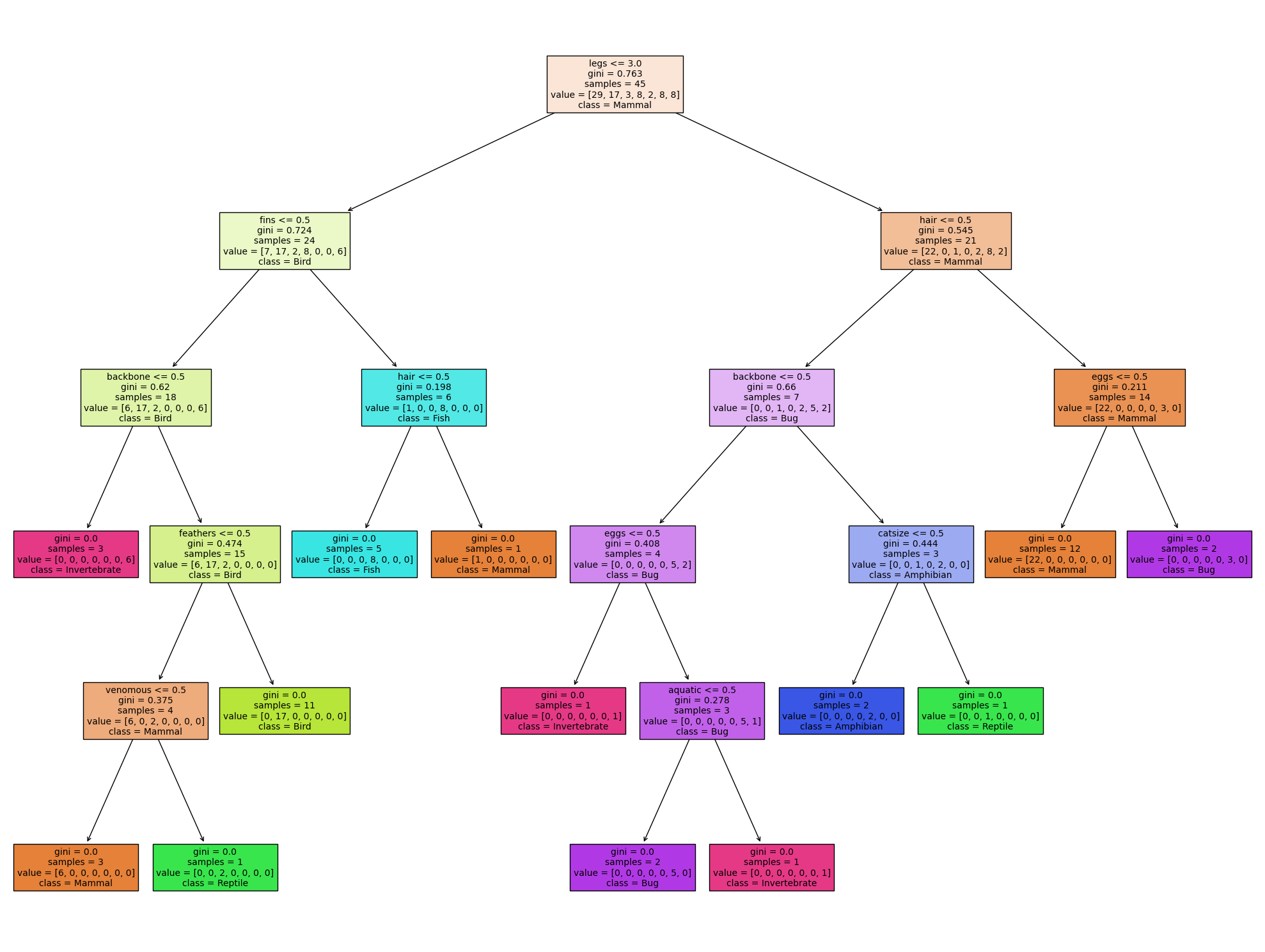
array(['Mammal', 'Bird', 'Reptile', 'Fish', 'Amphibian', 'Bug',  
 'Invertebrate'], dtype=object)

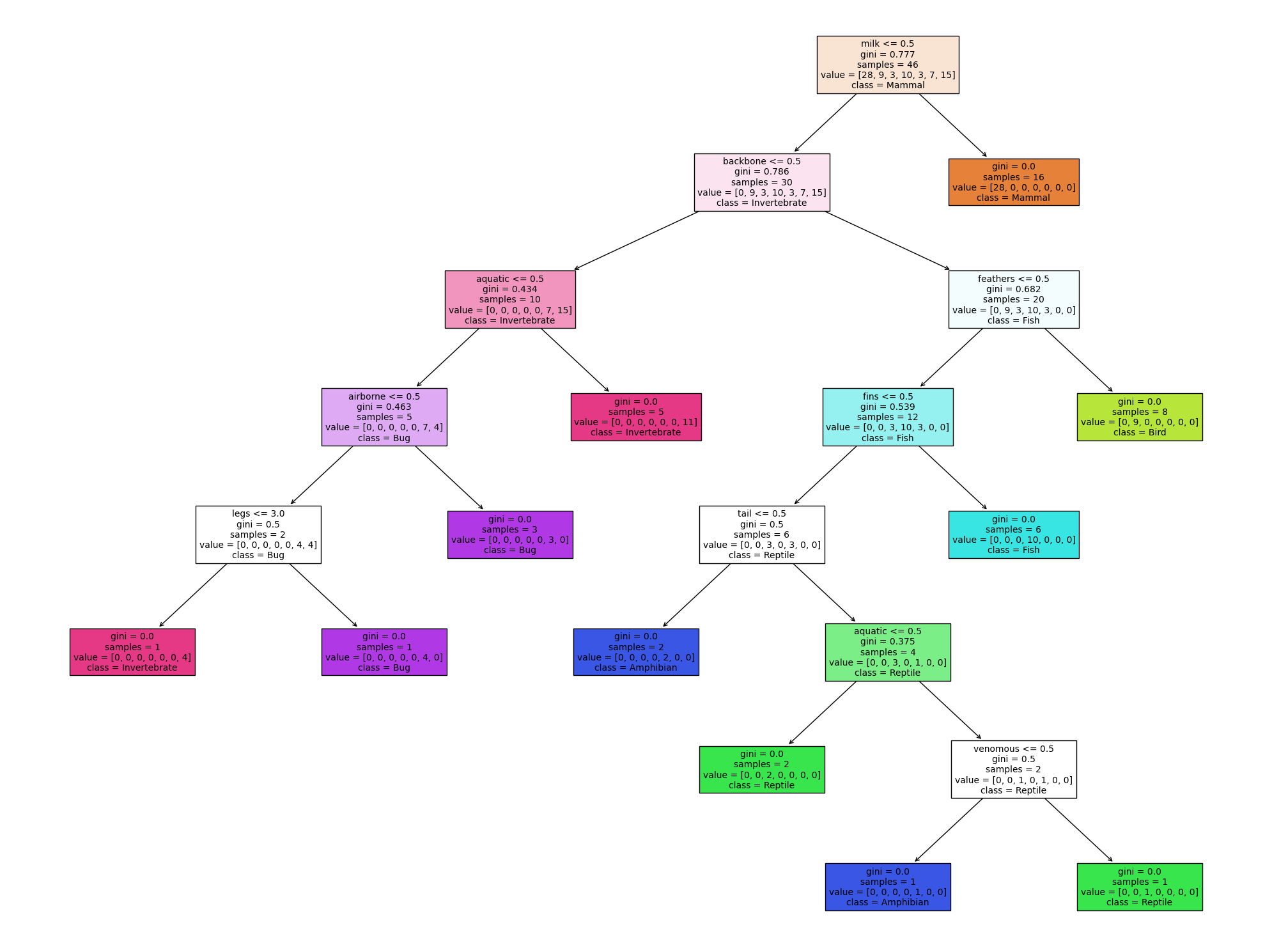
In [135]:

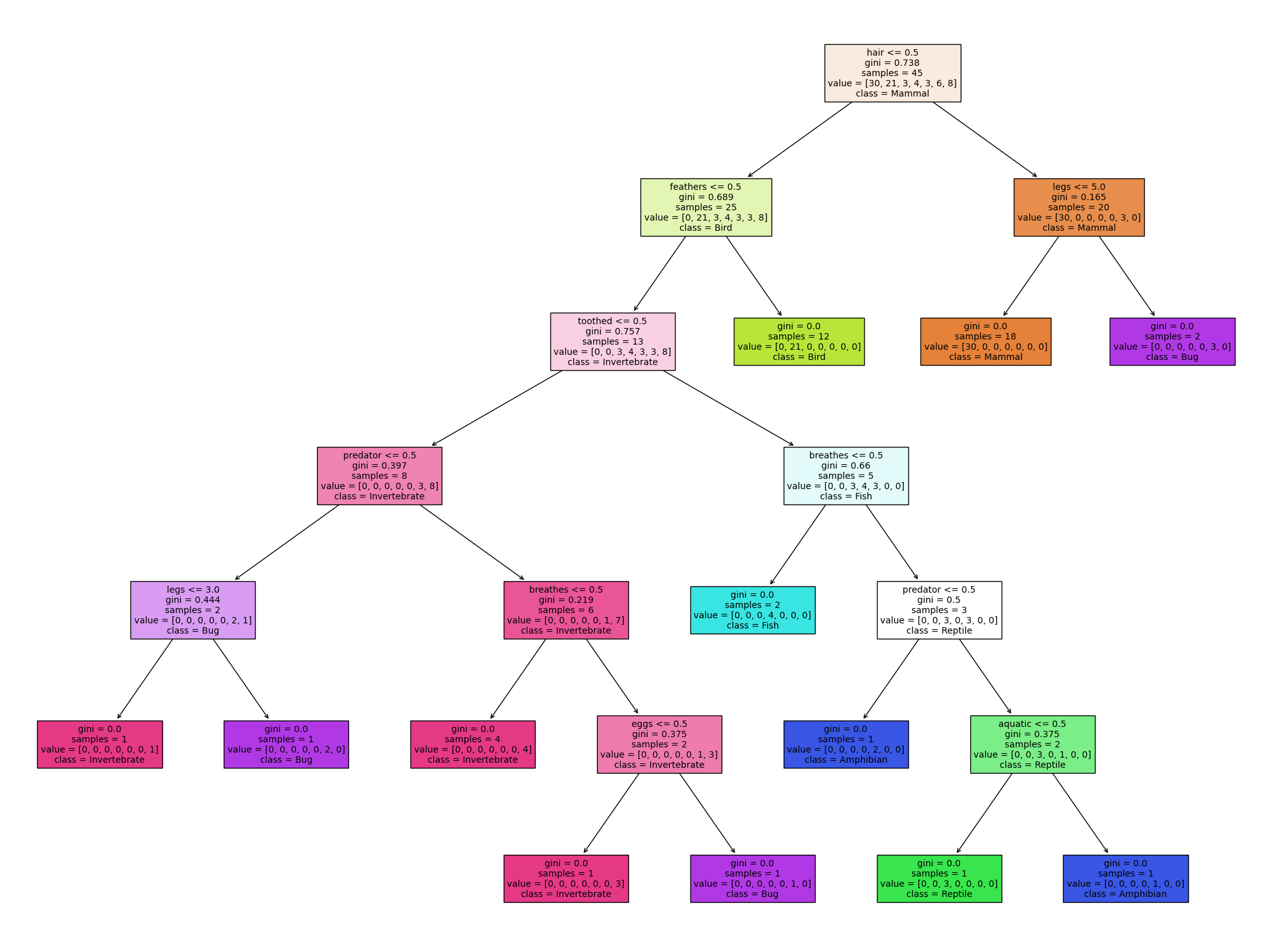
**from** sklearn.tree **import** plot\_tree  
  
for i **in** range(len(rf**.**estimators\_)):  
 plt**.**figure("Decison Tree", figsize **=** [20,15])  
 plot\_tree(rf**.**estimators\_[i], fontsize **=** 10, filled **=** **True**, feature\_names **=** features, class\_names **=** target)  
 plt**.**tight\_layout()  
 plt**.**show()

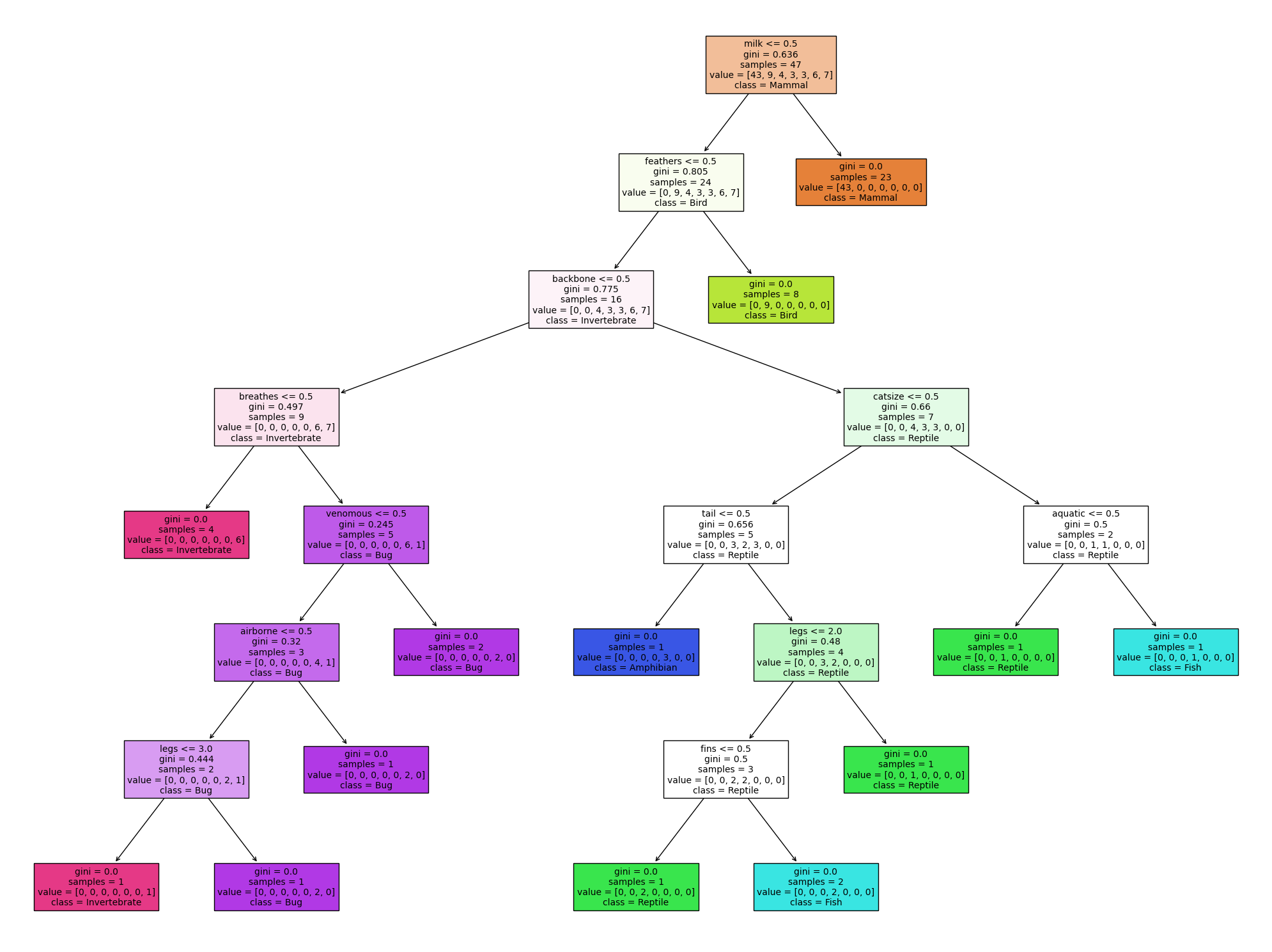


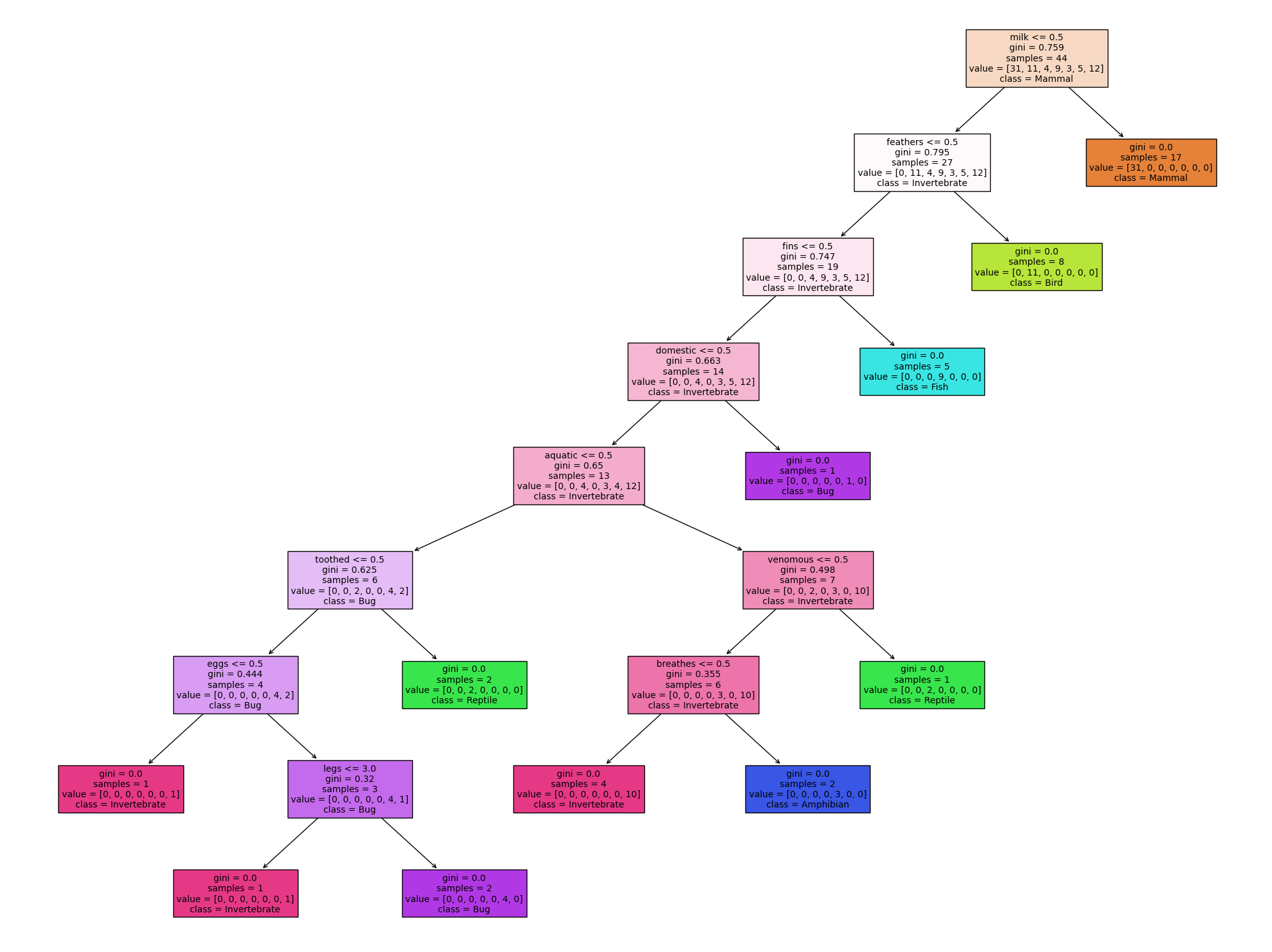


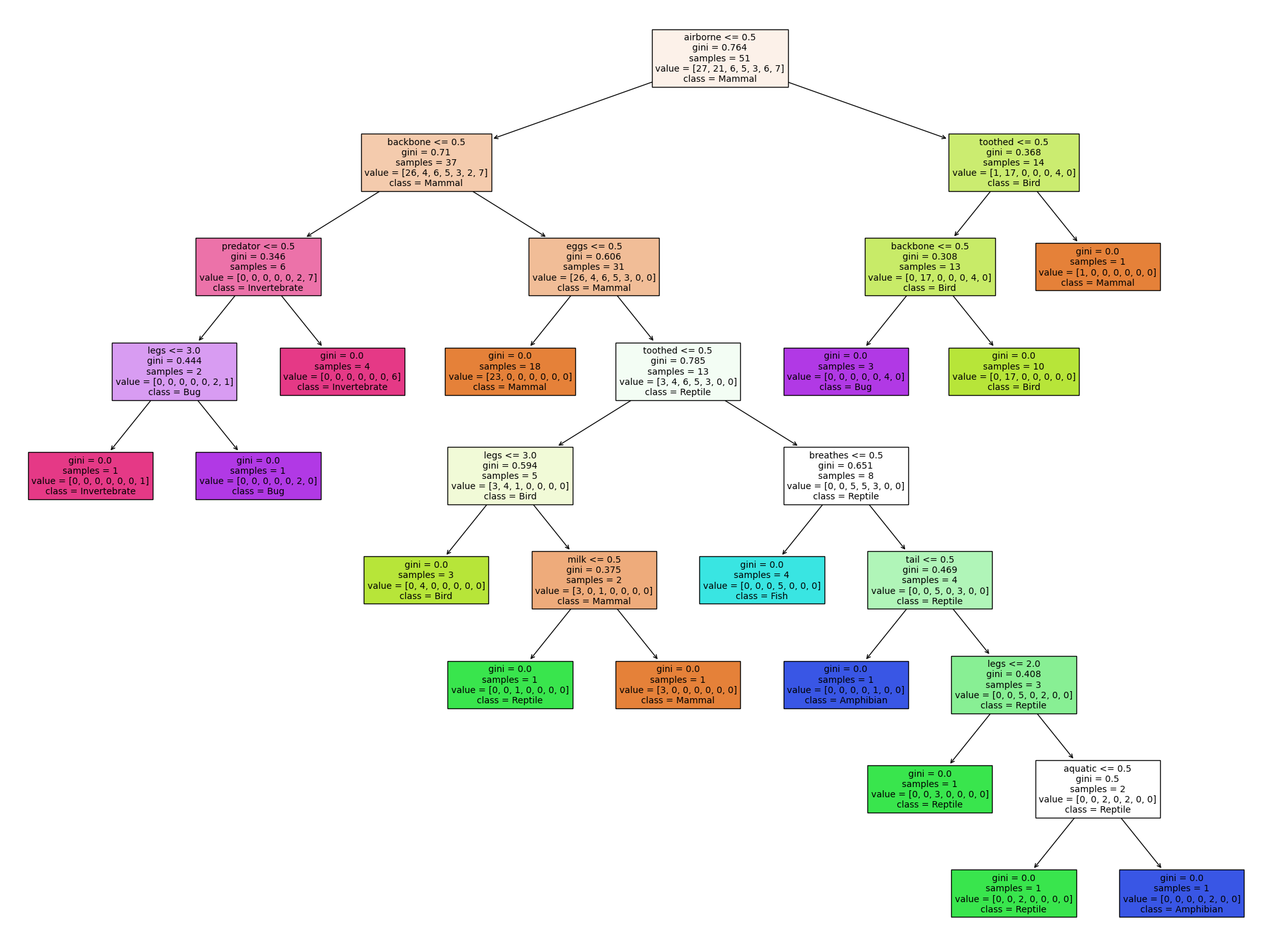


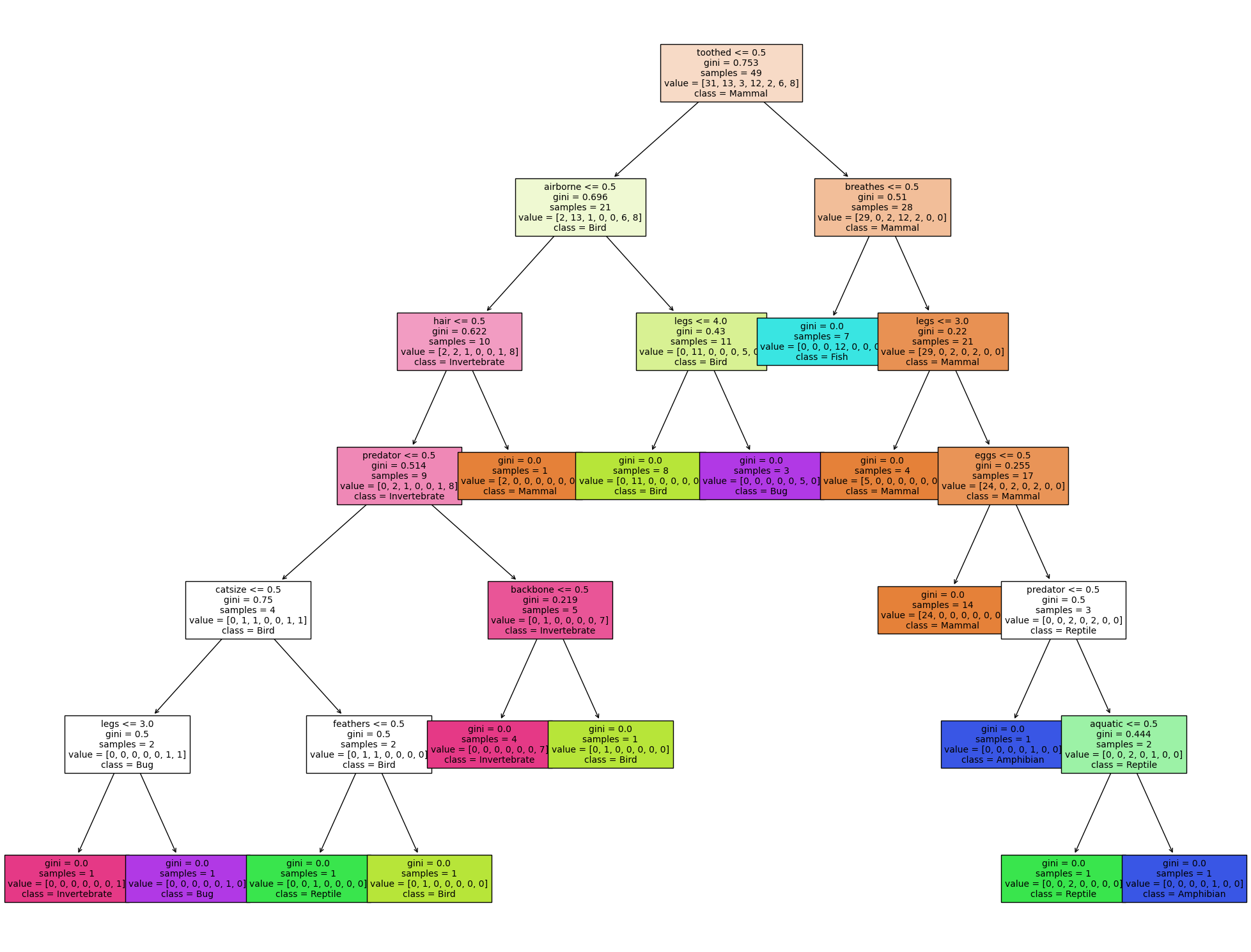


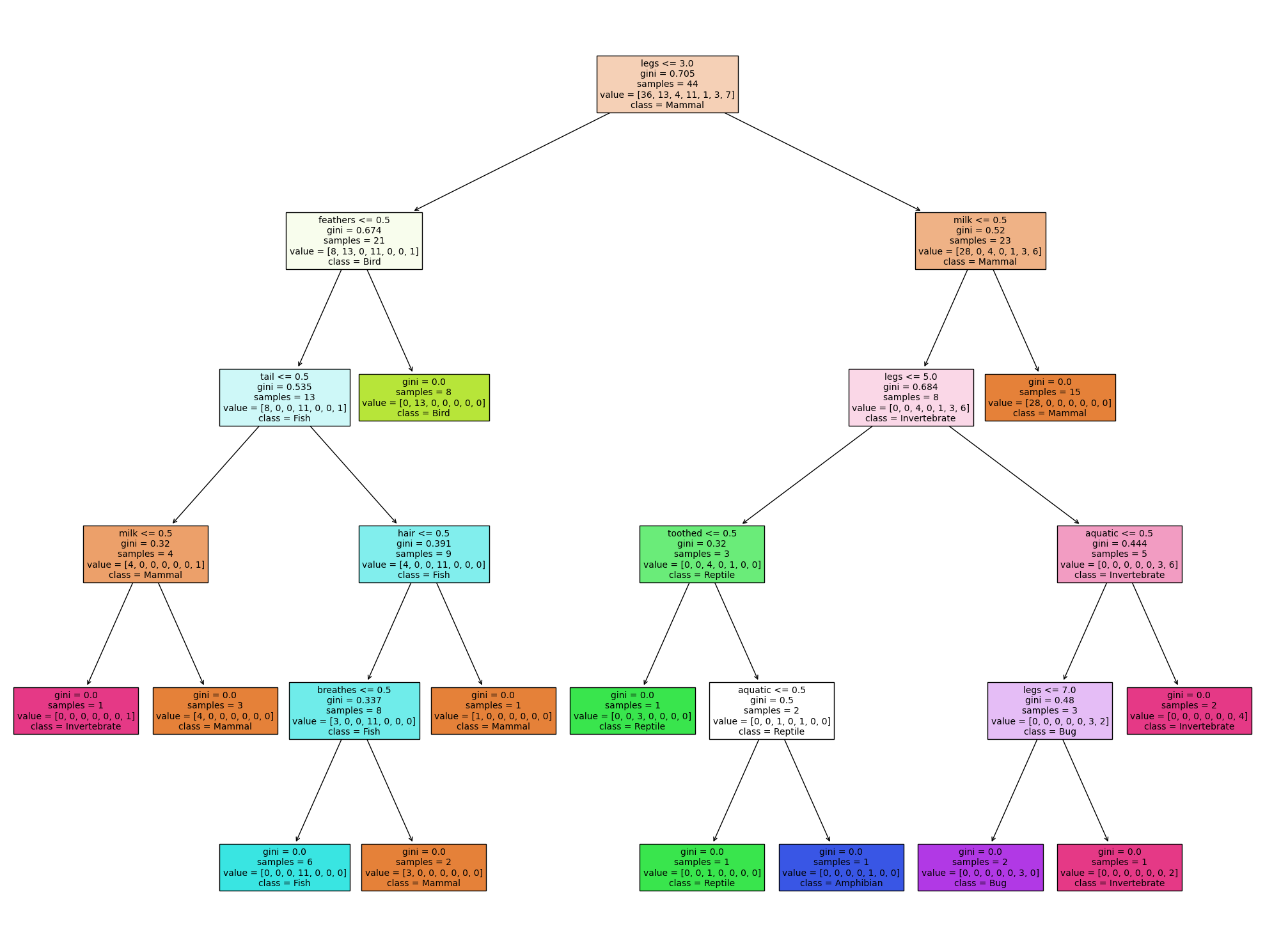












‌