Project Report

Tianwei Mo, tm3929
Ziwei Wang, zw3296

Quad chart

Motivation

As many as one third of all species are expected to go extinct this century, the knowledge of assessing threatened status of different species has become more and more serious. However, this assessment is a heavily manual process that requires highly expert knowledge.
Thus, our goal is to develop a machine learning that can automatically determine how at risk a species is. So the problem is a classification probems. Our model will take features abot the species as input and output a class label indicating the threatened status, from 'lc' (least concern) to 'ce' (critically endangered)

Innovation

Our innovation has 2 aspects. For one thing, we make up a python code file that can automatically crawl the data form the website to generate our dataset since the big species dataset online is not completed which only includes species traits. If we generate the dataset manually, we need to collect 4000 samples which is totally impossible.
For another thing, we focus on data pre processing, using several ways such as PCA to reduce the dimensions. We also use one-hot encoding to transform the class labels that cannot be recgnized.

Results

Before testing, we expect to reach 80% accuracy. The best model is logistic regression. The best accuracy is 67.33%. It did not reach to our expectation. Therefore, we failed the training.

Explanlation and Suggestions

The reason that our accuracy and loss is not very high is that our data samples is imbalance. The class label of 'lc' takes up the majority, which is almost 2300 samples, nearly 50%. The remaing is divided by other 5 labels, some of which only have 200 samples. That's why it's imbalance.
Thus, our first suggestion is that we need to reduce the amount of largest class label and generate more training samples for other labels. Secondly, our dataset has 57 features, some of which is useless such as species orders and sex. It's better to discard them.

1. Introduction

As the crisis of species extinction becomes more and more serious, knowledge of biogeography and macroecology is needed to help relieve the dilemma. Our goal of the project is to develop a machine learning method that can automatically determine how at risk a species is. Our models take information about the features of specie s as input (e.g. features related to their reproduction, diet. life habits, etc) and output a prediction indicating the threatened status. The models we used are SVM, BPNN(Back Propagation Neural Network), KNN(K-Neighbours), and RF(Random Forest)[1].

The data we mainly used came from COMBINE: a coalesced mammal database of intrinsic and extrinsic traits[2]. The dataset contains an extensive review of published mammal trait data sources between 1999 and May 2020. We used the latest version dataset (imputation_phylo_979.csv) for training.

IUCN Red List[3] records the current status of a wide range of species. According to the number of extant species, the website divides them into 7 species: least concern (LC), near threatened (NT), vulnerable (VU), endangered (EN), critically endangered (CR), extinct in the wild (EW), and extinct (EX). The website provides labels of the species from the COMBINE database.

2. Contribution

(1) Generate the dataset

We first download the dataset. While the dataset contains species and their feature, it does not include the threatened status, which is the labels. We wrote a python script (copy_data.py) to scrap the class labels from the IUCN website with the species names. However, the script performed quite slowly, eventually we searched for 4296 labels. Next, we joined the labels and features to form a complete dataset.

(2) Preprocessing

The raw dataset needs to be preprocessed. The preprocessing function first filled na value with the mean of the feature. There were two useless labels for training to be removed, "dd" for data deficient and "Remove" for no result. In addition, the function removed data with too few corresponding labels. We also made category features into one-hot features for better training results. This led to too many features, so we performed a PCA to select the main features, last the dataset was split into a training set and a testing set.

The total models we used are SVM, BPNN(Back Propagation Neural Network) , KNN(K-Neighbours) and RF(Random Forest). With the 2 methods we improved , we increased the average accuracy from 57% to 67%.

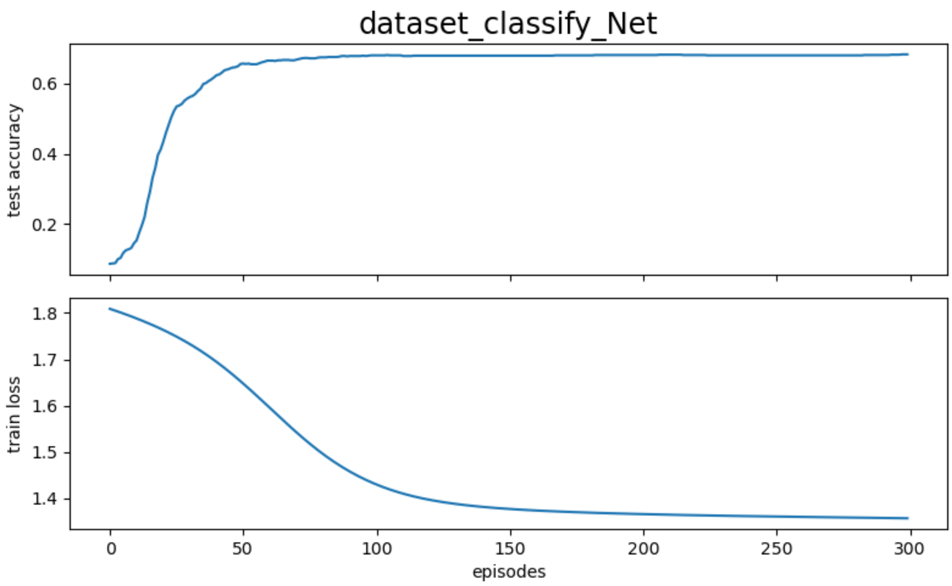
3. Training

Before training, we set a threshold of 80% for our models. We hope they could reach to on average 80% of accuracy.

For training, we used a back propagation neural network (BPNN). as the main model. Before training, we performed a cross-validating using logistic regression to find the optimal PC number. The total feature number is 57, and we select [10, 20, 25, 30, 40, 57] as the testing PC numbers. When PC number = 40, we got the best result:

```
Confusion matrix:
[[0.0033 0.01   0.     0.0295 0.     0.0014]
 [0.0028 0.013  0.     0.0902 0.     0.0041]
 [0.     0.     0.     0.0165 0.     0.0005]
 [0.0006 0.0088 0.0006 0.6441 0.     0.0078]
 [0.0006 0.0046 0.     0.0607 0.     0.0019]
 [0.0017 0.0107 0.     0.083  0.     0.0035]]
Mean:   0.6638418079096046
SE:     0.0032799373452889317
Best ncomp: 40
Highest accuracy: 0.6732580037664784
```

To train our BPNN, we first created a BPNN class. The BPNN had one hidden layer and an output layer with 30 hidden units in each hidden layer. The hidden layer used the RELU activation function and since this is a multi-class classification task, we used the softmax activation function for the output. In terms of the optimizer, after testing, we found that the Adam optimizer had the best performance. The training process contained 300 episodes and 50 episodes in each epoch. For each epoch, the training and testing accuracy and loss were printed on the console. At last, we got the highest accuracy of 68.07%.




```
The 0 training episodes, loss=1.8087645769119263, training accuracy=0.08434221148490906
accuracy on test set 0.08568738400936127
The 50 training episodes, loss=1.6475268602371216, training accuracy=0.6678773164749146
accuracy on test set 0.6572504639625549
The 100 training episodes, loss=1.4297831058502197, training accuracy=0.6775625348091125
accuracy on test set 0.6807909607887268
The 150 training episodes, loss=1.377063274383545, training accuracy=0.6811944842338562
accuracy on test set 0.6798493266105652
The 200 training episodes, loss=1.3659858703613281, training accuracy=0.6807909607887268
accuracy on test set 0.6817325949668884
The 250 training episodes, loss=1.3608072996139526, training accuracy=0.6840193867683411
accuracy on test set 0.6807909607887268
```

Apart from logistic regression and BPNN, to get the best result, we tried several other classification models, such as SVM, K-nearest neighbors (KNN), and random forest (RF). Generally speaking, they have similar performances.

We performed a simple SVM classifier to do the prediction. To select the best C and gamma, we did a cross-validation and found the best test score. It was the highest accuracy we had gotten (69.3%).

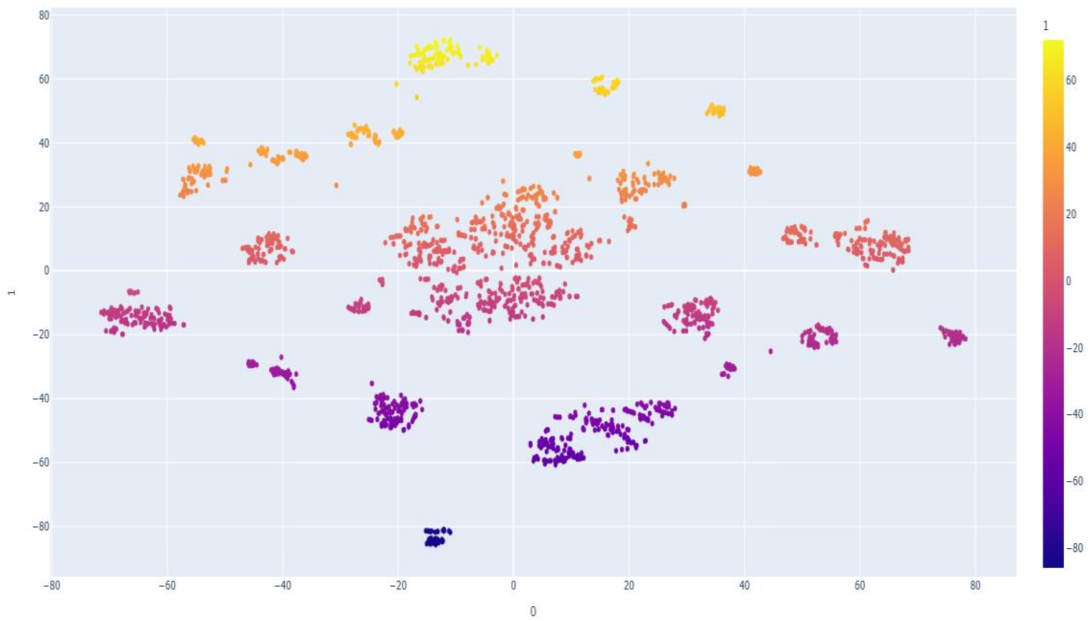
```
[CV 2/5; 9/9] START C=10, gamma=1, kernel=rbf.................................
[CV 2/5; 9/9] END C=10, gamma=1, kernel=rbf;, score=(train=0.944, test=0.647) total time=   1.7s
[CV 3/5; 9/9] START C=10, gamma=1, kernel=rbf.................................
[CV 3/5; 9/9] END C=10, gamma=1, kernel=rbf;, score=(train=0.941, test=0.654) total time=   1.7s
[CV 4/5; 9/9] START C=10, gamma=1, kernel=rbf.................................
[CV 4/5; 9/9] END C=10, gamma=1, kernel=rbf;, score=(train=0.944, test=0.655) total time=   1.7s
[CV 5/5; 9/9] START C=10, gamma=1, kernel=rbf.................................
[CV 5/5; 9/9] END C=10, gamma=1, kernel=rbf;, score=(train=0.942, test=0.657) total time=   1.7s
The best score is 0.6926553672316385
The best parameters are {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
```

Similarly, we built a simple KNN classifier with 6 neighbors since our samples were predicted to be one of the six classes. The plotted classification metrics showed that it did not perform better but similarly.

Precision:

	precision	recall	f1-score	support
0	0.24	0.15	0.18	47
1	0.30	0.22	0.25	117
2	0.30	0.33	0.32	18
3	0.74	0.93	0.82	703
4	0.14	0.01	0.03	72
5	0.24	0.08	0.12	105
accuracy			0.66	1062
macro avg	0.33	0.29	0.29	1062
weighted avg	0.57	0.66	0.60	1062

Before performing RF, we did a T-distributed stochastic neighbor embedding (TSNE) to reduce and visualize the data samples. Our code plotted a RF scatter graph to help us to visualize the clusters.



The feature importance matrix showed that all features were not quite relevant. Like KNN, printed the classification metrics. Their performance was quite similar.

```
Importance:  [0.03447232 0.02751171 0.04064192 0.02382048 0.02600243 0.02855172
 0.02154793 0.02360904 0.02595251 0.0228392  0.02526576 0.02098932
 0.02348485 0.02301491 0.02314193 0.02789369 0.02244852 0.02209453
 0.02517245 0.02649802 0.0206241  0.02412482 0.02196541 0.03315793
 0.03058773 0.02368075 0.02226009 0.0226412  0.02131627 0.02101872
 0.02331277 0.02270929 0.02664973 0.02740491 0.02383783 0.02222084
 0.02427962 0.02189206 0.02544372 0.02591898]
train: 0.9987893462469734
test: 0.6751412429378532 0.6751412429378532
Precision:
              precision    recall  f1-score   support

           0       0.17      0.09      0.11        47
           1       0.37      0.20      0.26       117
           2       0.50      0.22      0.31        18
           3       0.75      0.94      0.84       703
           4       0.32      0.10      0.15        72
           5       0.26      0.17      0.21       105

    accuracy                           0.68      1062
   macro avg       0.39      0.29      0.31      1062
weighted avg       0.60      0.68      0.62      1062
```

4. Performance Analysis

None of our models achieved our expectations. In general, they all have 67% accuracy and are not higher than 70%.

One of the potential problems could be the imbalance of data. We checked that among 3500 valid samples, 2300 of them had "lc" labels, and most of the other samples have about 100-200 samples. This resulted in high accuracy in predicting species with "lc" labels, and wrong results on other species. This can be visualized from the performance metrics, where label 3 always had an outstanding prediction accuracy.

Meanwhile, the low relevance of features affected the result. From the PoV table, we can see even the PC with the highest variance made little difference in samples. Besides, the importance matrix shows that all features have low importance, which means PCA did not help. From BPNN's accuracy plot we know that the network converged quickly. This indicates that the features did not provide much information to train.

```
PoV:
[0.17837512 0.28439995 0.35475302 0.40207964 0.43913502 0.46853363
 0.49332255 0.51675808 0.53866422 0.5598993  0.58014396 0.59998181
 0.61951042 0.63869162 0.65743706 0.67605883 0.69461132 0.71307056
 0.73147441 0.74981588 0.7681412  0.78644589 0.80473085 0.82299798
 0.84126142 0.85951503 0.87663245 0.89278397 0.90829362 0.92340621
 0.9359966  0.94673134 0.95631356 0.9651735  0.97353193 0.98041364
 0.98635166 0.99131378 0.99583565 1.        ]
```

Lastly, we used a limited number of samples since label fetching is time-consuming. There are about ⅓ samples that haven't been used.

5. Conclusion

To sustain human life and protect the ecosystem, we start the project to use machine learning models to predict species' endangered level. We improved the preprocessing process and tried several machine learning models and saw a performance enhancement from 57% accuracy to 69% accuracy. We discussed multiple issues. I hope our project made some contribution to protecting animal species diversity.

Reference:

[1]: Python机器学习笔记：随机森林算法 - 战争热诚 - 博客园 (cnblogs.com)

[2]: COMBINE: a coalesced mammal database of intrinsic and extrinsic traits - Soria - 2021 - Ecology - Wiley Online Library

[3]: IUCN Red List of Threatened Species