466 Project Report

Author: YiFan Wang

1. Data set:

Anuran Calls(MFCCs) Data Set

(Data source: https://archive.ics.uci.edu/ml/datasets/Anuran+Calls+%28MFCCs%29)

The data set is talk about the acoustic features extracted from syllables of anuran (frogs) calls, including the family, the genus, and the species labels (multilabel).

Data Set Characteristics:	Multivariate	Number of Instances:	7195	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	22	Date Donated	2017-02-24
Associated Tasks:	Classification, Clustering	Missing Values?	N/A	Number of Web Hits:	4490

From the graph above, we can see that my experiment has 7195 samples and 22 features. The type of the features is numerical and the target variable is the species.

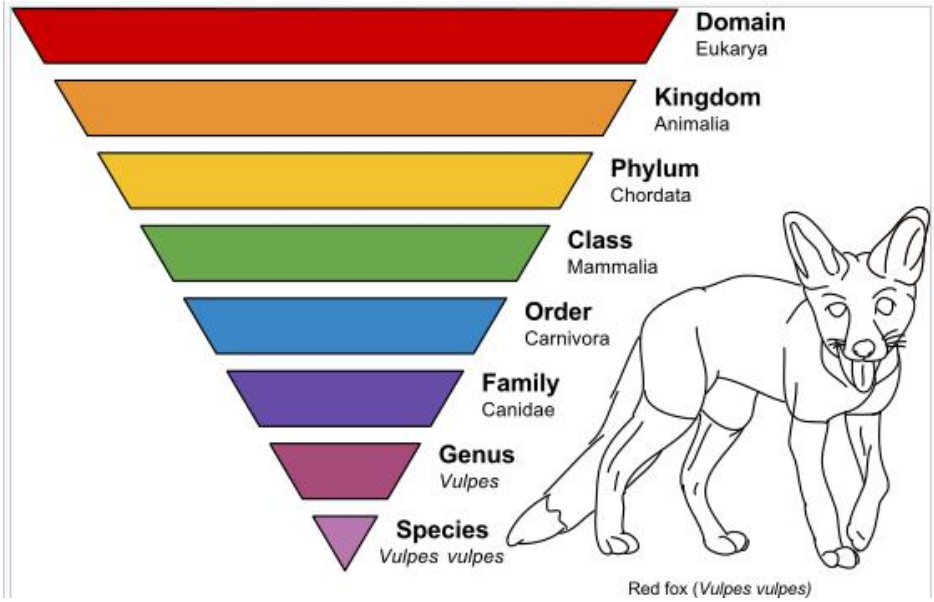
(1) Data Set Information:

This dataset was used in several classifications tasks related to the challenge of anuran species recognition through their calls. It is a multilabel dataset with three columns of labels. This dataset was created segmenting 60 audio records belonging to 4 different families, 8 genus, and 10 species. Each audio corresponds to one specimen (an individual frog), the record ID is also included as an extra column.

(2) Data construction:

It has 26 colums, it includes 22 columns of MFCCs variables, and it also includes the columns of family, genus, species label and the record ID. So they add up to 26 columns, I use pandas.read_csv to read the data directly.

Here , I would want to make some explanation for the column 23 to the column 25. They are the data for the family, genus and species of the frogs. I have found a graph to show the relationship among the family, genus and species from Wikipedia.



As the graph shown above, we can know that family>>genus>>species. And in my project, I only focus on the species. So I only pick the data of column 25 as the target variable, and I haven't used column 23 and column 24 as the feature values. Because in my experiment, we can only collect the data of MFCCs of the frogs, and then we can use the data to predict the species of it with my models. So I don't think that we can use the information of family and genus to predict it's species.

(3) Attribute Information:

In sound processing, the mel-frequency cepstrum is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an mel-frequency cepstrum (MFC). MFCCs are commonly used as features in speech recognition systems, but it used for species recognition in my experiment.

Due to each syllable has different length, every row (i) was normalized acording to MFCCs_i/(max(abs(MFCCs_i))).

2. Question:

(1). My question of this project:

If we have a set of data of MFCCs, which of these three models (LinearSVC, Naive Bayes and Logistic Regression) can provide us the most accurate result for species recognition?

(2). The importance of the problem:

The anuran species recognition is a challenging problem, because we can't see the difference among specimens sometimes. Then we have another method that we can get the audio records from their calls, and then we can use the models to predict the species of the specimens. So the data of the MFCCs was collected because they are the analytic data from the audio records.

3. Code:

-- There have three python files: main.py, dataloader.py and classalgorithms.py
I just simply used the code file from our assignment3 and modified it.

-- I have installed "sklearn" and "pandas" and use these two external libraries. "pandas" is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. And "sklearn" includes the machine learning algorithms. I just simply import it and use the model of the package to fit the data, and then predict from the test set.

-- In dataloader.py, use the preprocessing of sklearn to process the data, and then use the train_test_split without setting the testsize parameter to split the data into trainset and testset. So it has default testsize value of 0.25, so the trainset is triple of the size of testset.

-- In classalgorithms.py, I use three algorithms of sklearn to fit the data.

1. LinearSVC:

It's a model of support vector machines that is a set of supervised learning methods used for classification, regression and outliers detection. The reason that I use this algorithm is that it is effective in high dimensional space and it can use different kernel, because different kernel function can be used for the decision function.

2. Gaussian Naive Bayes:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The likelihood of the feature is assumed to be Gaussian, and I use it for Naive Bayes. I used it because naive bayes works well in many real-world situation, and it is faster than other methods.

3. Logistic regression(SGD):

I use it because it is efficient and it is easy for implementation, because it has lots of opportunities for code tuning. In the code file, I set loss="log" and penalty="l2" in SGDclassifier. So it will implement the logistic regression with L2 norm penalty. And I will set the loss equals to hinge to implement the linear support vector machine and set loss equals to modified_huber to implement smoothed hinge loss in the final writeup, so I can see which one is better to fit our data.
And there also have some reasons that why I choose L2 regularization in my project: L2 regularization has computational efficient due to having analytic solutions, and it has non-sparse outputs.

<u>The parameters that they tunned:</u>

It has two parameters that includes "regwgt" and "nh". I have set the values of "regwgt" to 0.0, 0.01, 0.05, 0.1, and I have also set the values of "nh" to 4, 8, 16, 32. In the output, it will report the error with different parameters, and my algorithm will also report the best parameter to fit every models.

<u>Result:</u>

	{"regwgt": 0.0, "nh":4}			{"regwgt": 0.01, "nh":8}			{"regwgt": 0.05, "nh":16}			{"regwgt": 0.1, "nh":32}		
run\model	LinearSVC	NaiveBayes	LogitReg	LinearSVC	NaiveBayes	LogitReg	LinearSVC	NaiveBayes	LogitReg	LinearSVC	NaiveBayes	LogitReg
1	5.34	9.62	7.28	5.34	9.62	7.05	5.34	9.62	7	5.34	9.62	7.34
2	5	9.23	6.78	5	9.23	6.95	5	9.23	6.84	5	9.23	7.06
3	4.89	8.45	6.34	4.89	8.45	6.39	4.89	8.45	6.61	4.89	8.45	6.56
4	5.39	9.28	6.84	5.39	9.28	6.73	5.39	9.28	6.73	5.39	9.28	6.78
5	4.78	8.23	7.17	4.78	8.23	7	4.78	8.23	7	4.78	8.23	6.95
6	4.65	8	6.61	4.65	8	6.5	4.65	8	6.5	4.65	8	6.56
7	4.72	9.06	6.5	4.72	9.06	6.5	4.72	9.06	6.56	4.72	9.06	6.84
8	5.23	9.51	7.06	5.23	9.51	7.12	5.23	9.51	7.06	5.23	9.51	7.12
9	4.61	8.56	6.45	4.61	8.56	6.5	4.61	8.56	6.56	4.61	8.56	6.39
10	4.67	7.95	6.5	4.67	7.95	6.45	4.67	7.95	6.28	4.67	7.95	6.17
Average error for Logistic Regression:			6.714841579									
Average error for LinearSVC:			4.930516954									
Average error for Naïve Bayes:			8.788215675									

As the graph shown above, there are the errors of different models with different parameters in 10 runs.

I think the “winner” algorithm of my experiment is the LinearSVC. First of all, it report the smallest error. Then, the data of my experiment has 7195 samples and 22 features, so it has high dimension. And LinearSVC is effective in high dimensional space.

Thank you for reading!

Source:

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
https://pandas.pydata.org/
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://archive.ics.uci.edu/ml/datasets/Adult
https://archive.ics.uci.edu/ml/datasets/Anuran+Calls+%28MFCCs%29
http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
http://scikit-learn.org/stable/modules/svm.html
http://scikit-learn.org/stable/modules/naive_bayes.html
https://en.wikipedia.org/wiki/Taxonomic_rank