The Uniqueness of Chinese Music by Data Mining Strategy

Name: Xiaoyu Qiao, Zhenhui Jiang

December 20, 2017

Abstract

This study will focus on building models to classify a group of music datasets into two classifications, Chinese music or non-Chinese music. As our subjective feelings, Chinese music is characteristic and easy to distinguish. Data mining strategy can objectively view Chinese music's uniqueness by the outcome of classification. Methods in supervised learning and unsupervised learning will be used. The classification methods will cover Logistic Regression, Random Forests, Boosting and shrinkage methods like Ridge Regression and Lasso Regression. Classification will be made based on audio features returned by default MARSYAS settings. Receiver Operating Characteristic(ROC) Curve will be used to compare the overall performance of different classification models, which indicates Ridge Regression has the best prediction. What's more, the most important features in the music dataset can be find during the Boosting and Random Forests Process. We conclude the computer can use shrinkage methods in regression to objectively distinguish the uniqueness of Chinese Music.

1 Introduction

Subjectively, Chinese music is characteristic because most of their rhythm are slow and are full of Chinese Folk Music such as bamboo flute, Zither and Erhu. The primary goal is to find whether the computer, objectively, distinguishes the uniqueness or not. The study will base on audio variables in the dataset to build and compare models to classify music data into different origins — China or non-China. By the results of models, the question that whether Chinese music is unique or not can be answered.

There's one paper, called "Predicting the Geographical Origin of Music" and published by Fang Zhou, Claire Q and Ross. D. King on magazine ICDM in 2014 trained a machine learning program to be able to predict the geographical origin of pieces of music. For our project, we focus on the classification and the performance in prediction of different classification methods.

The dataset is "Geographical Original of Music Data Set" obtained from UCI Machine Learning Repository. The dataset was built from a personal collection of 1059 tracks covering 33 countries/area. The program MARSYAS was used to extract audio features from the wave files. We used the default MARSYAS settings in single vector format (68 features) to estimate the performance with basic timbal information covering the entire length of each track. This dataset can assist to distinguish Chinese music and non-Chinese music.

By the data of area under ROC Curve(AUC), we notice that only Ridge Regression and Lasso Regression classification methods have AUC greater than 90%. They have better performance than other classification methods.

Based on our results, the uniqueness of Chinese music can be well-distinguished by computer, and the computer have good performance in deciding whether the origin of a piece of music is China or not when applying shrinkage classification methods.

2 Data and Methods

2.1 Data Preprocessing

The original dataset contains 70 columns: the first 68 columns represent 68 audio features from default MARSYAS settings in single vector format; the last two columns represent latitude and longitude respectively. `V1` to `V68` are audio features returned by default MARSYAS settings. The columns we are interested in are the latitude and the longitude, which will indicate the location of the music origin.

The `mutate` command will be used to add an attribute `ChinaOrNot` which indicates whether the origin is China or not. The original dataset has already been standardized. We use `complete.cases` command in R to remove all the missing values. Then there are 968 complete observations after the filtering. We set the training set to have 800 elements and the testing set to have 168 using random indexes generated by R.

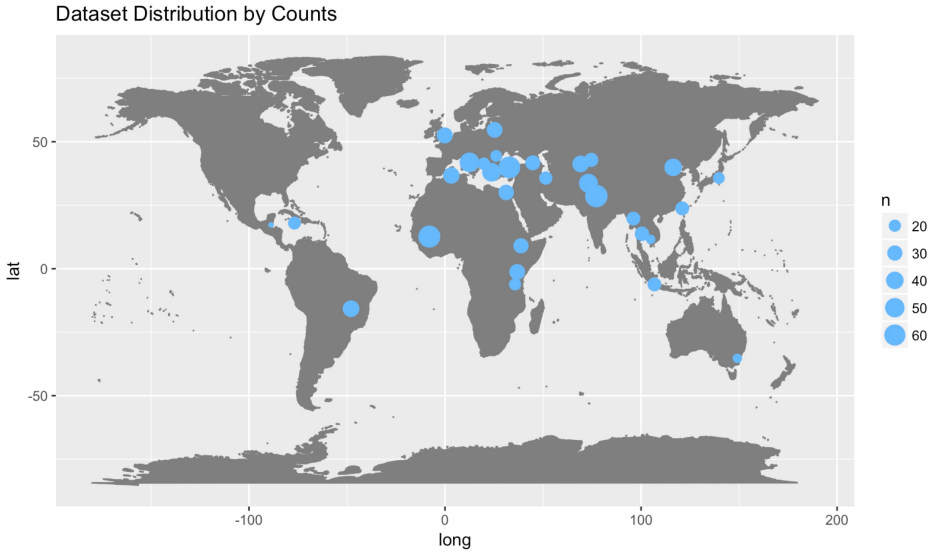
2.2 Data Visualization

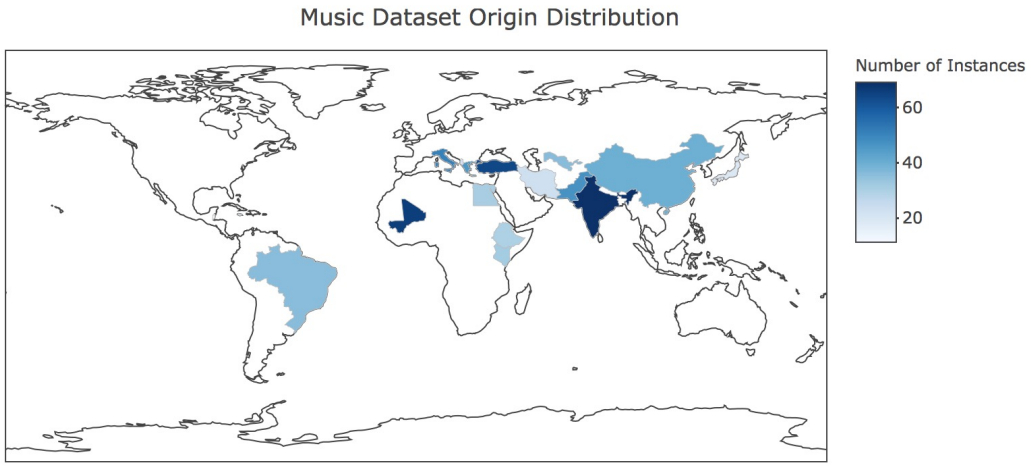
The `rworldmap` and `countrycode` packages in R will help identify the country and continent of each observation based on its latitude and longitude.

China	Non-China
40	928

Table 1. The Counts of Music data's Origins

To visualize the data as a whole, we use `ggmap` and `plotly` packages to plot the geographical distribution of the data set.





Notice that the Chinese music data are mainly collected from Chinese Ethnic minority autonomous regions like Xinjiang, Tibet, and Gangxi. This implies a variety of ethnic minority characteristics in the targeted Chinese music, and those music pieces usually have more emphasis on traditional Chinese musical instruments, which will very likely influence some timbal variables in audio features extracted by MARSYAS.

2.3 Data Mining

To test the music-origin detecting performance, we will use multiple classification methods. We will have a first look on their prediction performance at the confusion matrix. In the ROC Curve section, there will be a comparison for TRP and FPR at all thresholds(i.e. ROC Curve) between different classification models.

2.3.1 Decision on Threshold

In order to create the confusion matrix for test data, we want to control False Negative Rate and False Positive Rate to be as small as possible at the same time. We'd like to choose probability threshold that is closest to $(FPR, FNR) = (0, 0)$. In our case, for each classification method, we calculate the euclidean distance between each point of (FPR, FNR) and $(0, 0)$, and print probability threshold with the smallest euclidean distance as `best`.

Note: in our cases, we will find out the best threshold for each model, thus we are using different thresholds for producing different models' confusion matrix.

2.3.2 Logistic Regression

In our case, we have to use Binary classification, with response equal "China" and "non-China". To build a Logistic Regression classification, we use `gbm()` to fit generalized linear model and `summary()` to produce result summaries of various model fitting functions.

Then we construct confusion matrix for the testing data. We use `type ="response"` in `predict()` function to get a sequence of probabilities.

2.3.3 Random Forest

We apply bagging and random forests to the music data, using `randomForest()` function can be used to perform both random forests and bagging. And the function `varImpPlot()` can be used to show the most important variables based on Model Accuracy and Gini value. We also calculate the test error rate for comparing with boosting method.

2.3.4 Boosting

The `gbm()` function will fit boosted classification trees to the data set. We firstly transform response variable(country) to be coded as $0, 1$ instead of two levels. We also set the option distribution = `bernoulli` to fit the binary problem.

The `summary()` function produces a relative influence plot and also outputs the relative influence statistics, which will indicate the most important variables.

The partial dependence plots for most important variables illustrate the marginal effect of the variables on the response after integrating out the other variables.

2.3.5 Ridge Regression and Lasso Regression

Different from logistic regression, Shrinkage methods regularize estimates to reduce variance by shrinking coefficients toward 0. All predictors are kept, but we will constrain complexity of model fit. Two important regressions are ridge regression and lasso regression.

The ridge or lasso regression model is fitted by calling the `glmnet` function with $\alpha = 0$ and $\alpha = 1$ respectively, which returns a sequence of models to choose from. `cv.glmnet` is the main function to do **cross-validation** here, which could also do plotting and prediction. `plot()` will return the whole path of coefficients(the option `xvar` has various choices such as `"dev"` and `"lambda"`). `coef()` will return the coefficient vector corresponding to the best model by cross-validation. And `cv.lasso$lambda.min` is corresponding to the λ that gives minimum mean cross-validated error.

Different from ridge regression, lasso regression will do variable selection.

2.3.6 ROC Curve

We use package `ROCR` to generate ROC curve, which indicates the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. We use function `predict` to predict the outcome(orgin of music) based on audio features `V1` to `V68`, and functions `prediction` and

`performance` to build the ROC Curve. We will use AUC to compare the **OVERALL** performances of different classification methods.

3 Results

3.1 Logistic Regression

First, we look at the output of the summary of logistic regression (in Appendix). Only 10 audio features(`V3`, `V5`, `V20`, `V37`, `V40`, `V42`, `V52`, `V54`, `V58`, `V60`) are statistically significant at level 0.01.

Below is the confusion matrix for the `best` threshold = 0.01229

	true	
pred	No	Yes
No	128	2
Yes	33	5

Table 2. Confusion Matrix for Logistic Regression on Test Data

– Out of 168 cases, the model classifies 128 + 5 = 133 correctly (79.17%)
– Out of 161 non-Chinese music, the model classifies 128 correctly (79.5%)
– Out of 7 Chinese music, the model classifies 5 correctly (71.73%)

The Truth Positive Rate is 71.73% at the best threshold for Logistic Regression.

3.2 Random Forest

By the summary of the Random Forest model (in Appendix), the number of variables tried at each split is 8 and the out-of-bag error is 4.12%.

Below is the confusion matrix for the `best` threshold = 0.052

	true	
pred	No	Yes
No	125	1
Yes	36	6

Table 3. Confusion Matrix for Random Forest on Test Data

– Out of 168 cases, the model classifies 125 + 6 = 131 correctly (77.8398%)
– Out of 161 non-Chinese music, the model classifies 125 correctly (77.64%)
– Out of 7 Chinese music, the model classifies 6 correctly (85.71%)

The test error rate obtained is 0.04167. This indicates a good performance over bagging in this case.

The Variable Importance Plot below has a decreasing order of importance based on Model Accuracy and Gini value.

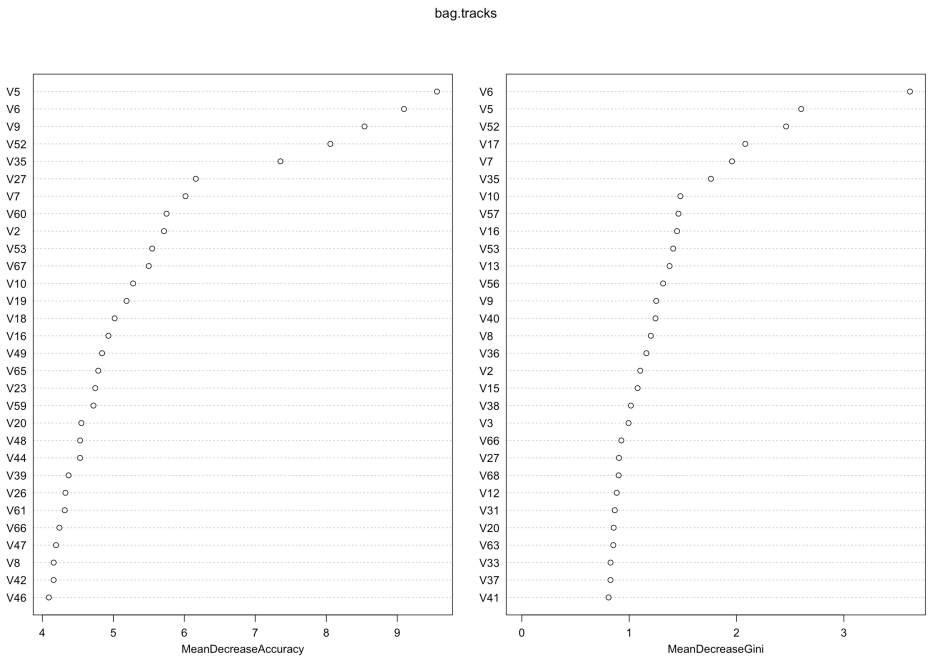


Figure 2. The Variance Importances by Random Forest Method

The results indicate that across all of the trees considered in the random forest, `V5` and `V6` are by far the most important variables in terms of Model Accuracy and Gini index.

3.3 Boosting

Below is the confusion matrix for the `best` threshold = 0.02815

	true	
pred	No	Yes
No	137	1
Yes	24	6

Table 4. Confusion Matrix for Boosting on Test Data

– Out of 168 cases, the model classifies 137 + 6 = 143 correctly (85.12%)
– Out of 161 non-Chinese music, the model classifies 137 correctly (85.09%)
– Out of 7 Chinese music, the model classifies 6 correctly (85.71%)

Notice that Boosting has worse performance than Random Forest since its test error rate is 0.994.

The summary output below gives the importance order or variables.

	var <fctr>	rel.inf <dbl>
V5	V5	21.47314
V6	V6	19.70719
V7	V7	6.06705
V35	V35	6.03707
V57	V57	5.52813
V40	V40	4.22049
V10	V10	4.04129
V17	V17	3.78227
V39	V39	2.91125
V9	V9	2.89748

Table 3. Variable Importance of Boosting (Top 10)

We see that `V5` and `V6` are by far the most important variables, which coincides with the results of Random Forest.

The partial dependence plots for `V5` and `V6` shows what marginal effect they have on `ChinaOrNot` after integrating out the other variables.

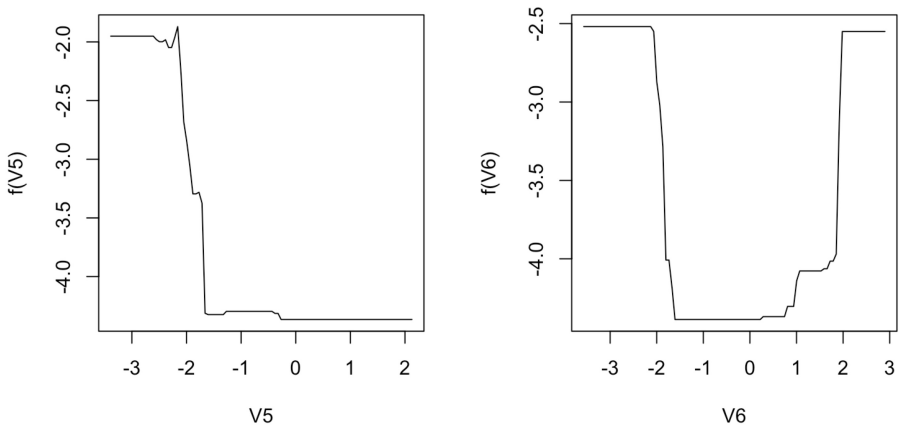


Figure 3. The partial dependence plots for `V5` and `V6`

3.4 Ridge Regression

Since the above classification methods are not satisfactory, we try shrinkage classification methods by constraining complexity of model fit.

We start with Ridge Regression.

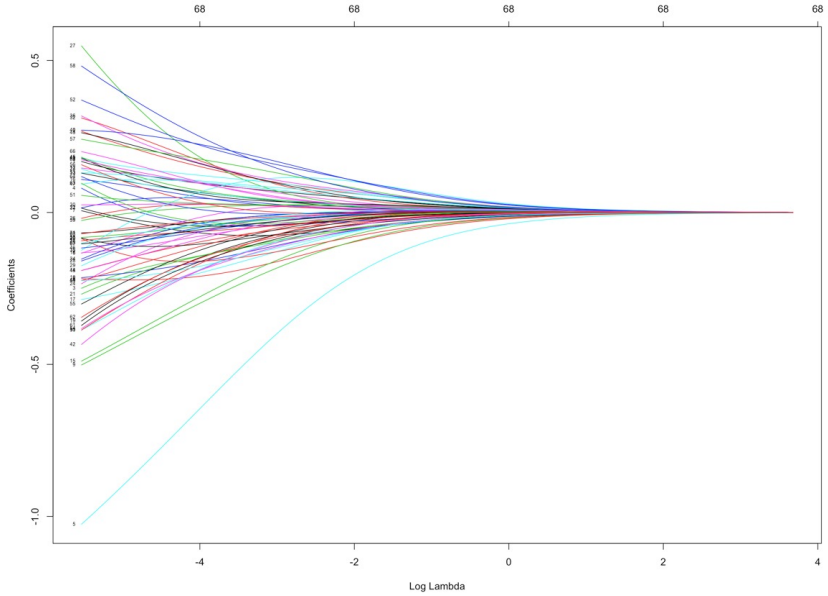


Figure 4. Ridge Regression Lambda

The above plot is as a function of log lambda. When log lambda is 2, all the coefficients are essentially zero. When reducing lambda, the coefficients get bigger from zero smoothly. When the log lambda is getting close to -6, the sum of squares of coefficients keeps growing.

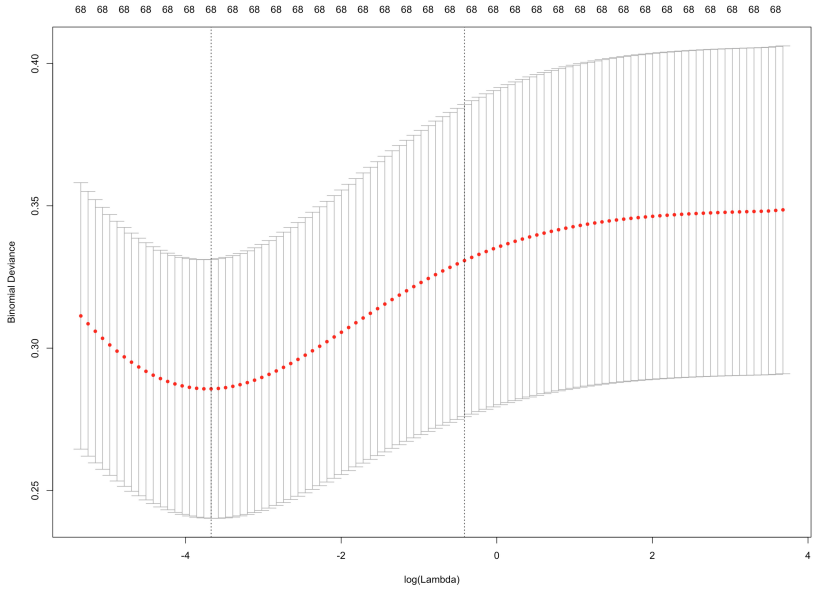


Figure 5. Ridge Regression Model Selection

The above is a whole path of model given by Ridge regression. The red dotted line is the cross-validation curve and there are error bars. Two vertical dotted lines are respectively λ that gives minimum mean cross-validated error and λ that gives the most regularized model such that error is within one standard error of the minimum.

We pick `lambda.min`, and below is the confusion matrix for the `best` threshold = 0.08549.

	true	
pred	No	Yes
No	147	1
Yes	14	6

Table 5. Confusion Matrix for Ridge Regression on Test Data

– Out of 168 cases, the model classifies 147 + 6 = 153 correctly (91.07%)
– Out of 161 non-Chinese music, the model classifies 147 correctly (91.3%)
– Out of 7 Chinese music, the model classifies 6 correctly (85.71%)

3.5 Lasso Regression

Different from Ridge Regression, Lasso Regression can do variable selection.

In the plot, the x-axis shows the percentage of deviance explained.

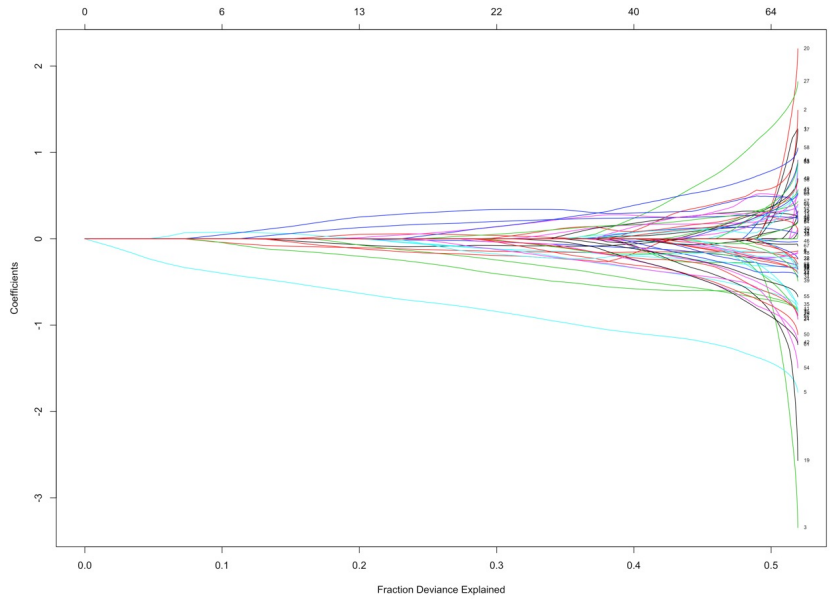


Figure 6. Lasso Regression Fraction Deviance Explained

In the end of the x-axis with a relatively small increase in deviance explained from between 0.48 and 0.52, coefficients become very large, which means the end of the path is over-fitting.

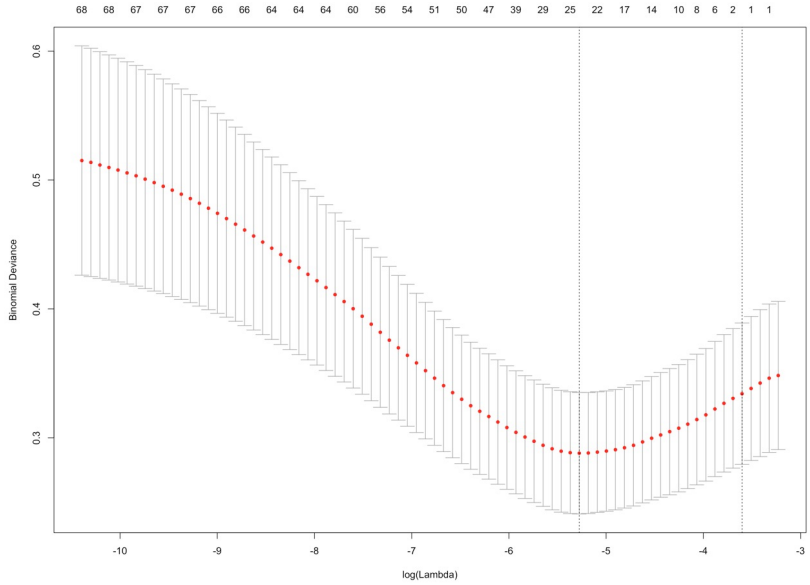


Figure 7. Lasso Regression Model Selection

The axis above indicates the number of variables. Our best model returned by `cv.glmnet` with `"lambda.min"` has 24 coefficients(not including intercept), which means the best model in Lasso Regression model math have 24 variables(see the coefficient output in Appendix).

Below is the confusion matrix for the `best` threshold = 0.04973.

	true	
pred	No	Yes
No	132	1
Yes	29	6

Table 6. Confusion Matrix for Lasso Regression on Test Data

– Out of 168 cases, the model classifies 132 + 6 = 138 correctly (82.14%)
– Out of 161 non-Chinese music, the model classifies 132 correctly (81.99%)
– Out of 7 Chinese music, the model classifies 6 correctly (85.71%)

3.6 ROC Curve

Now, to check those models' overall performance, we plot the following ROC Curve:

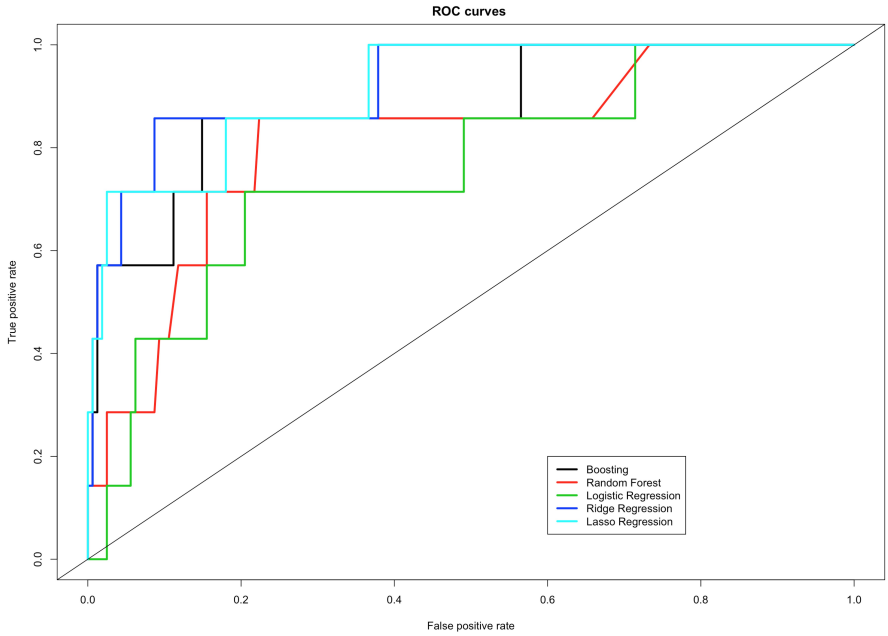


Figure 7. ROC Curves for All Classification Models

And their AUC are:

Logistic Regression	Random Forest	Boosting	Ridge Regression	Lasso Regression
0.756	0.8146	0.8776	0.9237	0.9148

Table 7. AUC results of classification models

It is clear that Ridge Regression and Lasso Regression have AUC greater than 90%, and Ridge Regression has the best OVERALL performance in predicting whether the origin of one piece of music is China or not.

4 Discussion

After the classification modeling process, we successfully single out one classification model — Ridge Regression, who can well-predict the uniqueness of Chinese music. Even it's not perfectly fitting the actual results, it still have the best performance among all other methods we pick.

The first big happened in the process of plotting the geographical distribution of the dataset on the world map. Since this is not part of what we learned in class, it took us a long time to actually figure it out. The second big challenger happened in the interpretation of plots for shrinkage methods.

Our primary conclusions are: the computer can objectively distinguish the uniqueness of Chinese music using the 68 audio features extracted from MARSYAS. The Ridge Regression performs very well in the predicting process, based on the comparison of different models' area under ROC Curves in our analysis step.

Some of the classifications look very unsatisfactory. This may due to the relatively small size of the dataset(only 968 instances). We are interested in obtaining a larger dataset(over 10,000 instances) from various resources, and try to find how classification may be done on the new dataset.

5 Miscellaneous

This dataset "Geographical Origin of Music Data Set" is obtained from UCI Machine Learning Repository. The main software used is R programming. We also use "plotly" package to plot the geographical distribution map.

The online resources including introduction on shrinkage methods from **Gerardnico** website, Ridge Regression Analysis from Data Scientist MR. Ricardo Carvalho's blog, "Glmnet Vignette"

by Mr. Trevor Hastie and Mr. Junyang Qian from Stanford Academic Blog, as well as PSTAT 131 lecture notes on Piazza.

The book references include *An Introduction to Statistical Learning* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *Elements of Statistical Learning* by Trevor Hastie, Robert Tibshirani, and Jerome Friedman, as well as *R for Data Science* by Garrett Grolemund and Hadley Wickham.

This study is directed by Professor Alexander Franks. The statements in this paper only represent the author's opinion.

References

[1]Franks, A., *Lecture 5 - Classification with Logistic Regression.* Lecture presented at PSTAT 131 Class in BUCHANAN HALL, GOLETA.

[2]Franks, A., *PSTAT 131/231: Regularization.* Lecture presented at PSTAT 131 Class in BUCHANAN HALL, Goleta.

[3]Hastie, T. J., Tibshirani, R. J., & Friedman, J. H. (2019). *The Elements of Statistical Learning.* Springer.

[4]Hastie, T., & Qian, J. (2014, June 26). Glmnet Vignette. Retrieved December 12, 2017, from https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

[5] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: With Applications in R.* New York: Springer.

[6]R - Shrinkage Method (Ridge Regression and Lasso). (n.d.). Retrieved December 19, 2017, from https://gerardnico.com/wiki/lang/r/ridge_lasso

[7]Wickham, H., & Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* Beijing: OReilly.

Appendix

Summary output for logistic regression:

```
Call:
glm(formula = ChinaOrNot ~ ., family = binomial, data = tracks_train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.800  -0.085  -0.020  -0.004   3.357

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -8.4560     1.4768   -5.73  1.0e-08 ***
V1            2.1768     3.0685    0.71    0.478
V2            1.5778     3.0339    0.52    0.603
V3           -6.9398     3.1807   -2.18    0.029 *
V4            0.4797     0.9456    0.51    0.612
V5           -1.9682     0.4992   -3.94  8.1e-05 ***
V6           -0.0865     0.6810   -0.13    0.899
V7            0.6326     0.4978    1.27    0.204
V8           -0.1158     0.4051   -0.29    0.775
V9           -0.1868     0.4205   -0.44    0.657
V10          -0.6318     0.4306   -1.47    0.142
V11          -1.0174     0.5574   -1.83    0.068 .
V12           0.2159     0.4269    0.51    0.613
V13           0.4075     0.3828    1.06    0.287
V14          -0.1954     0.4629   -0.42    0.673
V15          -0.7707     0.4166   -1.85    0.064 .
V16          -0.1870     0.3891   -0.48    0.631
V17          -0.2006     0.3725   -0.54    0.590
V18          -1.9925     3.1868   -0.63    0.532
V19          -0.8231     2.9804   -0.28    0.782
V20           3.9839     1.8106    2.20    0.028 *
V21          -1.5379     1.0027   -1.53    0.125
V22          -0.0872     0.7234   -0.12    0.904
V23          -0.5487     0.8333   -0.66    0.510
V24          -0.4257     0.7314   -0.58    0.561
V25           1.0659     1.0199    1.05    0.296
V26          -2.4456     1.2757   -1.92    0.055 .
V27           0.8441     0.9626    0.88    0.381
V28           1.1631     1.1131    1.04    0.296
V29          -1.0206     1.0198   -1.00    0.317
V30           0.5739     1.0464    0.55    0.583
V31           0.1478     0.9684    0.15    0.879
V32           0.1062     1.0369    0.10    0.918
V33           0.1295     1.0971    0.12    0.906
V34          -0.0579     0.9630   -0.06    0.952
V35          -2.1936     1.1570   -1.90    0.058 .
V36           0.2839     1.4662    0.19    0.846
V37           2.3793     1.1103    2.14    0.032 *
V38          -0.4870     0.5172   -0.94    0.346
V39          -0.5005     0.7260   -0.69    0.491
V40           1.2158     0.5157    2.36    0.018 *
V41           0.4613     0.4485    1.03    0.304
V42          -1.8012     0.8187   -2.20    0.028 *
V43           1.0424     0.5732    1.82    0.069 .
V44          -1.0672     0.6161   -1.73    0.083 .
V45           0.5741     0.6108    0.94    0.347
```

```
V46          -0.4937     0.6574   -0.75    0.453
V47           0.0605     0.5959    0.10    0.919
V48          -0.6364     0.8357   -0.76    0.446
V49           1.0859     0.7641    1.42    0.155
V50          -0.5011     0.7742   -0.65    0.517
V51          -0.6408     0.6740   -0.95    0.342
V52           2.8817     1.2538    2.30    0.022 *
V53           0.0572     1.1644    0.05    0.961
V54          -2.3146     0.9316   -2.48    0.013 *
V55          -1.3016     1.2355   -1.05    0.292
V56           0.8865     0.5572    1.59    0.112
V57           0.2655     0.4788    0.55    0.579
V58           1.5555     0.6731    2.31    0.021 *
V59          -0.8262     0.6550   -1.26    0.207
V60           1.7577     0.7754    2.27    0.023 *
V61          -0.7118     0.7980   -0.89    0.372
V62          -1.1550     0.9363   -1.23    0.217
V63           0.5127     0.7733    0.66    0.507
V64           0.1123     0.7186    0.16    0.876
V65          -0.1705     0.7772   -0.22    0.826
V66           0.3809     0.9888    0.39    0.700
V67          -0.7504     1.0041   -0.75    0.455
V68           0.4820     0.8773    0.55    0.583
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 275.03  on 799  degrees of freedom
Residual deviance: 115.60  on 731  degrees of freedom
AIC: 253.6

Number of Fisher Scoring iterations: 10
```

Random Forest Output:

```
Call:
 randomForest(formula = ChinaOrNot ~ ., data = tracks_train, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 8

        OOB estimate of  error rate: 4.12%
Confusion matrix:
     No Yes class.error
No  767   0           0
Yes  33   0           1
```

The Coefficient Output of Best Model in Lasso Regression Model Selection:

```
69 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept) -3.88798
V1           .
V2           0.02209
V3           .
V4           .
```

```
V5          -0.92050
V6           .
V7          -0.08169
V8          -0.15029
V9          -0.46863
V10         -0.05505
V11         -0.17280
V12          0.03426
V13          .
V14          .
V15         -0.30590
V16          .
V17         -0.13718
V18          .
V19          .
V20         -0.21325
V21         -0.07069
V22          .
V23          .
V24          .
V25          .
V26          .
V27          .
V28          .
V29          .
V30          .
V31          .
V32          0.07939
V33          .
V34          .
V35          .
V36          0.19345
V37          .
V38          .
V39          .
V40          0.21788
V41          .
V42         -0.05295
V43          .
V44          .
V45          .
V46          .
V47          .
V48          .
V49          .
V50         -0.12187
V51          .
V52          0.34022
V53          .
V54         -0.18756
V55          .
V56          .
V57          0.08965
V58          0.19949
V59         -0.21346
V60          .
V61         -0.01438
V62         -0.06351
```

```
V63             .
V64             .
V65             .
V66             .
V67             .
V68             .
```