# Qeexo Coding Challenge Report

Xin XU Email Linkedin GitHub

11/8/2018

## Contents

1. INTRODUCTION ..... 1

2. FEATURE EXTRACTION ..... 1

3. EDA(EXPLORATION DATA ANALYSIS) ..... 1

4. MODEL TRAINING ..... 6

5. SOME THOUGHTS: ..... 8

NOTES ..... 8

# 1. Introduction

This is a classification problem. Given the audio information, touch information(touch location, size, pressure, etc.) and the surface type(hand or table), we need to make a prediction about the gesture(knuckle or pad).

# 2. Feature Extraction

I import the **utility.py** which has several useful functions help me to load data, load labels and writing test results.

For touch information features, dropped the ‘pressure’ and ‘orientation’ who have only one single value.

For acoustic features, use FFT method to extract 154 features(including 129 bands, down-sample 129 bands into an additional vector of length 16 (i.e., buckets of 8), average absolute amplitude, total absolute amplitude, standard deviation of the absolute amplitude, and center of mass for both the segmented input signal and the FFT (8 features), estimate the fundamental frequency)

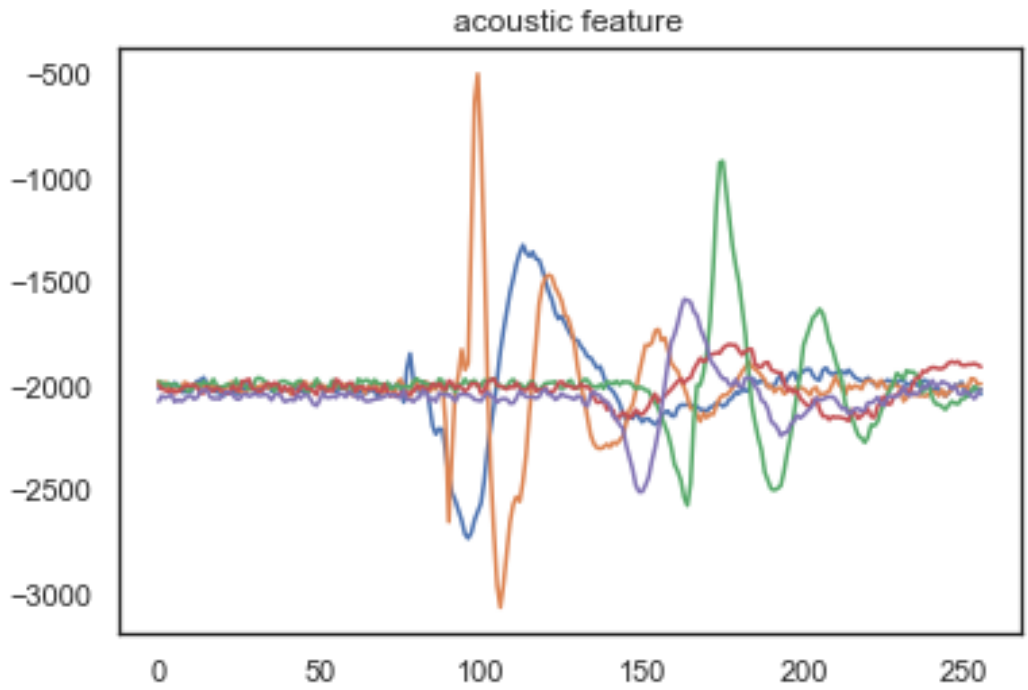
Why better than MFCC?

Intuition: in MFCC, after doing Fourier Transform, it would map the spectrum onto the Mel scale. The Mel scale is good for human listeners. We don’t need this step here. And the map of Mel scale may lose some potentially useful information.

# 3. EDA(Exploration Data Analysis)

1. How does the wav file look like?

   I randomly picked 5 instances to plot the audio file. The plots are as follows:

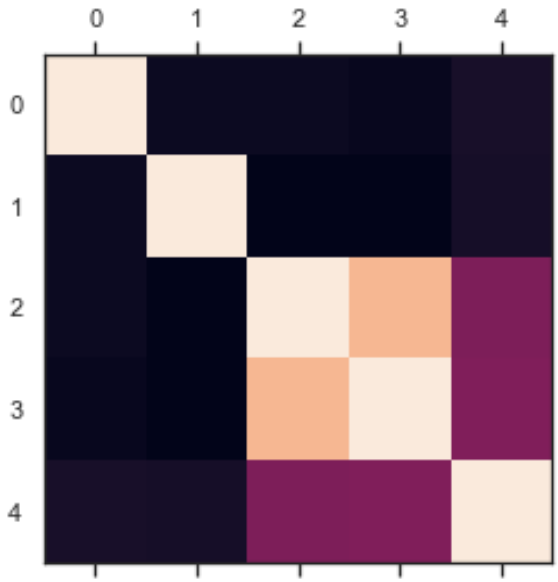



2. Are the touch and surface types features all useful?

| | x | y | major | minor | pressure | orientation | HoldingStatus |
|---|---|---|---|---|---|---|---|
| **count** | 20659.000000 | 20659.000000 | 20659.000000 | 20659.000000 | 20659.0 | 20659.0 | 20659.000000 |
| **mean** | 541.317440 | 940.129145 | 4.578586 | 4.223099 | 0.0 | -1.0 | 0.420737 |
| **std** | 289.368943 | 502.680342 | 1.013873 | 1.006628 | 0.0 | 0.0 | 0.493689 |
| **min** | 0.000000 | 0.000000 | 2.000000 | 2.000000 | 0.0 | -1.0 | 0.000000 |
| **25%** | 290.000000 | 497.000000 | 4.000000 | 4.000000 | 0.0 | -1.0 | 0.000000 |
| **50%** | 542.000000 | 938.000000 | 4.000000 | 4.000000 | 0.0 | -1.0 | 0.000000 |
| **75%** | 791.500000 | 1379.000000 | 5.000000 | 5.000000 | 0.0 | -1.0 | 1.000000 |
| **max** | 1079.000000 | 1919.000000 | 11.000000 | 10.000000 | 0.0 | -1.0 | 1.000000 |

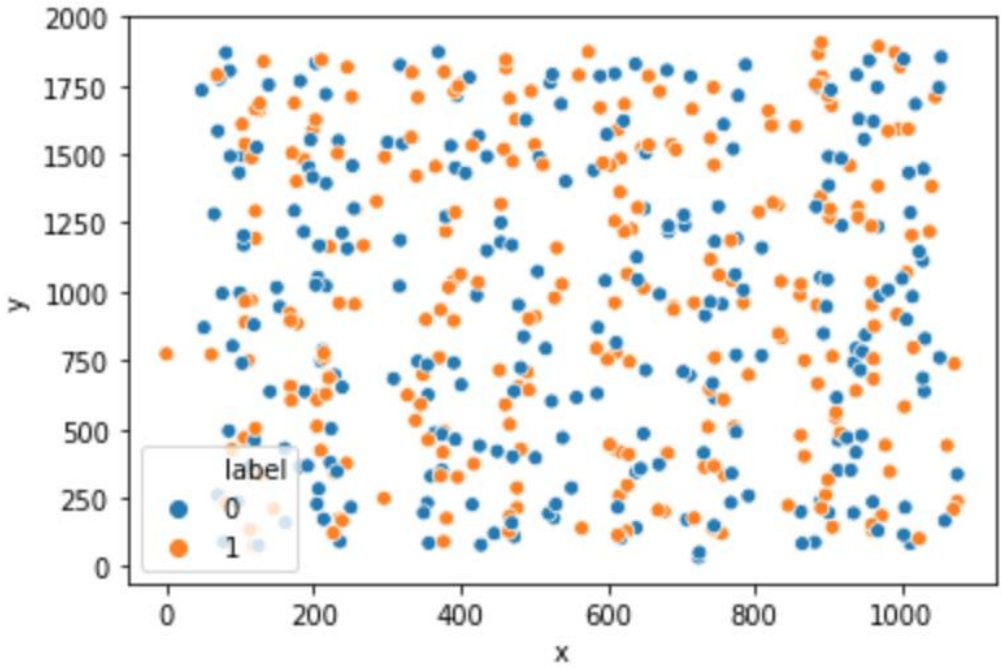
From the statistics summary, I see that column **pressure and orientation** are single values(with std equals 0) which are useless for our analysis, I dropped them.

3. Correlation between touch and surface type features?



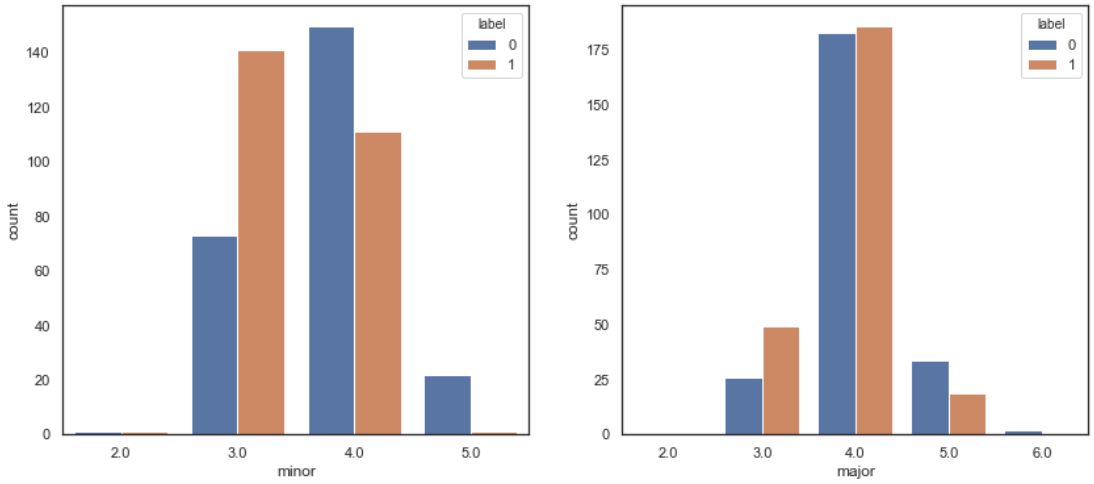

From the heatmap I see that feature 2 and 3(‘major’ and ‘minor’) have relatively stronger relationship.

4. Is there any pattern about the event location with regard to different labels?



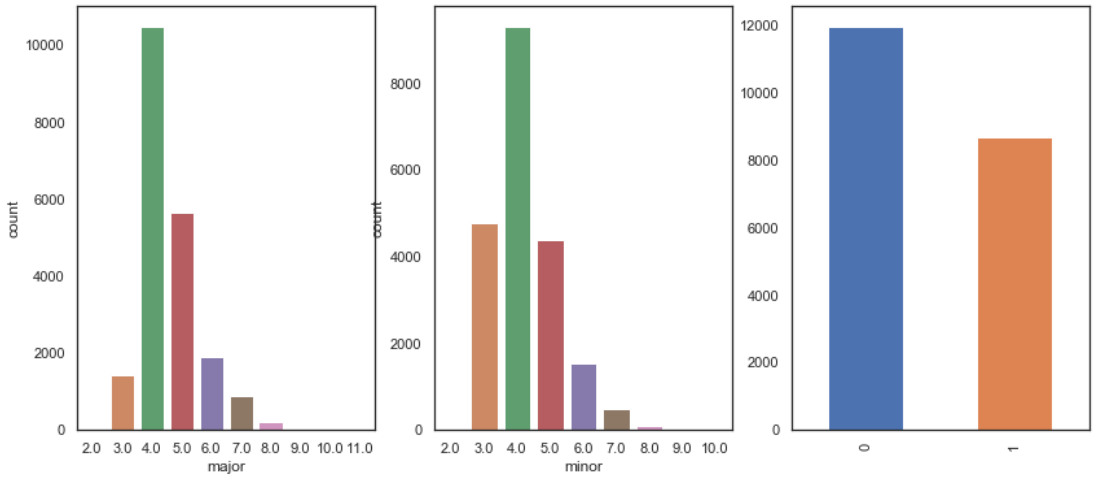

From the plot(scatter plot based on first 500 instaces), I can’t see any patter or relationship between location and labels

5. Is there any pattern about the event size with regard to different labels?

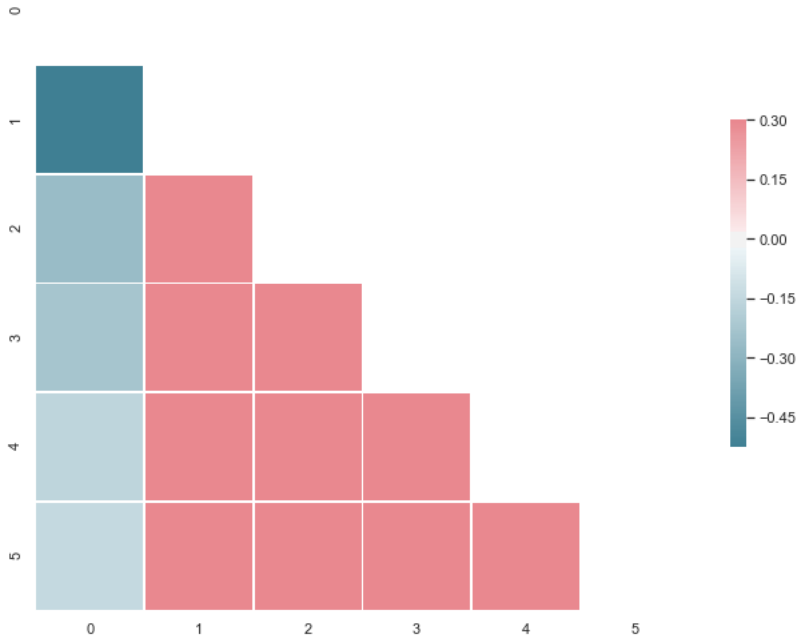



There might be some relationship between minor and label, but not very obvious.

6. How does these features(**major, minor, and surface type**) distributed?



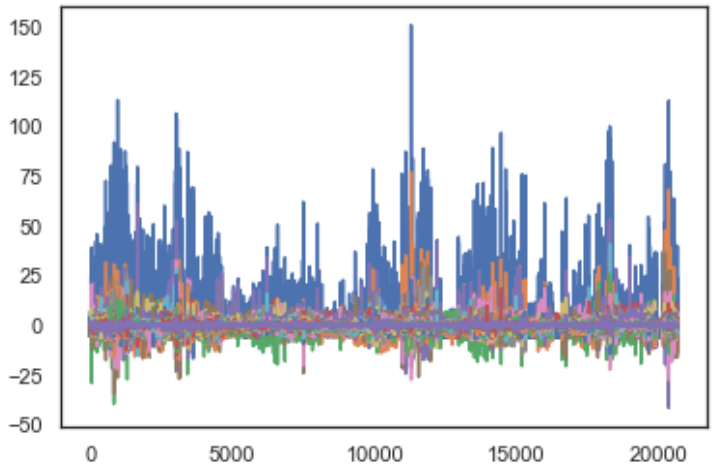

7. Check the first five audio features' correlation




8. PCA or not?

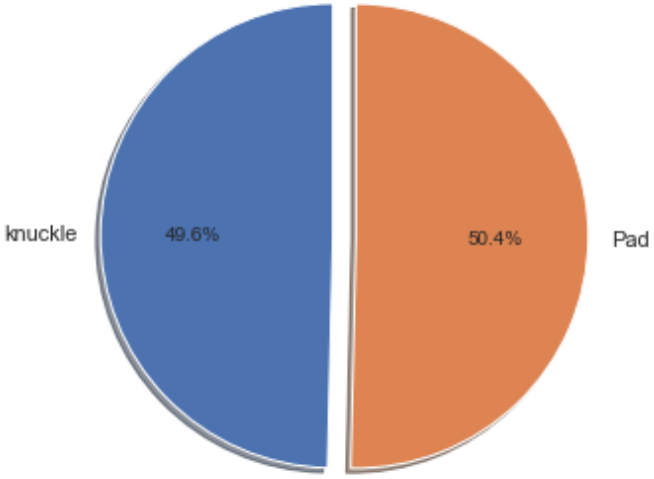
   As I extracted 154 audio features, it's too many to analyze, I applied PCA to reduce the audio features and after that I only have 35 audio features which still explained 0.95 variance. However, we should still compare the models that with PCA and without PCA. Later, we would see that without PCA is a little bit better than doing PCA.

9. How does the audio features(after PCA) distribution look like?



10. Are the labels balanced?

    It’s very balanced data in terms of the labels




## 4. Model Training

1. Split Training data into two groups: training data, and Testing Data
2. Scale data to reduce outliers
3. Random Forest(after grid search) showed 0.9501 out of bag error.
4. SVM(after grid search) showed similar accuracy with 0.95(cross validation result) and 0.9523 on the testing dataset.

```
cross validation score
[0.95251059 0.95190563 0.94765507 0.94856278 0.94916793]
cross validation accuracy: 0.9500 (+/- 0.00)

AUC Score (Test): 0.952348
Testset report
             precision    recall  f1-score   support

          0       0.96      0.95      0.95      2112
          1       0.95      0.95      0.95      2020

avg / total       0.95      0.95      0.95      4132

Testset confusion matrix
[[2009  103]
 [  94 1926]]
Testset accuracy
0.952323
```

5. Xgboost(after grid search) showed 0.9627 accuracy(cross validation result) and 0.9699 on the testing dataset.

```
cross validation score
[0.96249244 0.9646098  0.96187595 0.96187595 0.96248109]
cross validation accuracy: 0.9627 (+/- 0.00)

AUC Score (Test): 0.969975
Testset report
             precision    recall  f1-score   support

          0       0.97      0.97      0.97      2112
          1       0.97      0.97      0.97      2020

avg / total       0.97      0.97      0.97      4132

Testset confusion matrix
[[2050   62]
 [  62 1958]]
Testset accuracy
0.96999
```

The best for Xgboost by using grid search is with parameters (booster = 'gbtree', learning_rate = 0.3, max_depth = 6)

6. I chose this fitted model to predict the real Test dataset and output the result to file.

## 5. Some Thoughts:

Where can I change to improve the model?

I would see feature extraction. In the feature extraction part, After FFT, we have some features which are complex numbers. If directly use it as model input, the model will discard the imagine part which obviously would lose some information. I tried to regard one complex number feature as two features(real part and imaginary part) instead, the model performance is bad as well. I also tried to use absolute value and phase as the features. The performance is fine. Then I transferred the complex number into its absolute value(length). The model showed a better results.

However, I believe there could exist a better way to fix this problem. By doing this, we should have a deep domain knowledge about the acoustic and complex number input for SVM.

## Notes

In the given file utility.py, I made 1 change:

1. In the write_results function, I changed the “wb” to “w” in the “with open”function