Philippine e-Journal for Applied Research and Development 9(2019), 40-48 ISSN 2449-3694 (Online)
http://pejard.slu.edu.ph/vol.9/2019.10.11.pdf

Feature Selection for Heartbeat Classification using Naïve Bayes and K-Nearest Neighbor

Claizel Coubeili L. Cepe[1] and Maria Art Antonette D. Clariño[1*]

[1]*Institute of Computer Science, College of Arts and Sciences, University of the Philippines Los Baños, 4031 Los Baños, Laguna*

* *Corresponding author (mdclarino@up.edu.ph)*

Received, 13 March 2019; Accepted, 18 September 2019; Published, 11 October 2019

Abstract

This study determined which feature set can best classify heartbeat auditory data using Gaussian Naïve Bayes and K-Nearest Neighbor classifiers. Different feature sets (Low-level, Mel-frequency Cepstral Coefficients, or their combination) were used with the selected machine learning techniques. Results were evaluated using both micro- (for each class) and macro-averaging (across all classes) of precision, recall, and f-score. The balanced accuracy of the trials for each feature set was also measured. Two data sets used (Set A contains 84 data, and Set B with 432 data) were processed separately with a partition of 80:20 (training:testing). For data set A, Naïve Bayes with MFCC feature set garnered the highest macro-averages of recall (40%) and balanced accuracy (38.9%), while the same method with combined feature sets resulted in the highest precision (57.3%) and f-score (42.3%). For data set B, combined feature sets used on Naïve Bayes resulted in the highest macro-averages of precision (62.69%) and balanced accuracy (49.97%), while KNN with low-level feature set resulted in the highest recall (61.54%) and f-score (59.38%). The results show that Naïve Bayes and feature set 3 garnered the highest macro-averages because the combination of the low-level and MFCC features worked well with statistical approach. In the case of KNN as an unsupervised learning method, creating clusters from identified similarities was easier with low-level characteristics.

Keywords

Machine learning techniques for classifying heartbeat, feature extraction from auditory data, multi-class metrics, feature set comparison, LibROSA

Introduction

Current technology plays a vital role in the early and automatic detection of heart diseases through the processing of a patient's heartbeat. Several studies have classified heartbeats into categories using features extracted from different types of data like images or sounds. Some studies have extracted features from electrocardiogram (ECG) using Linear Prediction Modeling (Lin & Yang, 2014) and Time-Frequency feature extraction (Herrero et al., 2005). Another used Phonocardiogram (PCG) signals to classify normal and abnormal sounds in a heartbeat (Singh & Cheema, 2013).

Different types of features have also been explored in developing classifiers. For instance, Kumar et al. (2010) generated four feature sets (i.e., including all features, time-domain features, frequency domain features, and statistical domain features) to classify heart murmur into three common types (i.e., Mitral Stenosis, Aortic

insufficiency, and Mitral insufficiency). Their study focused on identifying which among the defined feature sets will perform best. After extracting the features, training, and applying a support vector machine (SVM) classifier, they reported that the best results were achieved using the superset of features.

Apart from the features, selection of method to use in the classification is also important. Based on Mierswa and Morik (2005), linear SVM classified the audio recordings better than K-Nearest Neighbor and Naïve Bayes. Their study evaluated the results by measuring classification errors, precision, and recall of these classification methods. Another study classified the heartbeat signals using several techniques, namely: Bayes Network, Naïve Bayes, Stochastic Gradient Descent, and Logist Boost (Singh and Cheema, 2013). Out of these methods, Naïve Bayes was found to be the most suited for the application, having the highest accuracy.

Most studies on heartbeat classification extracted features from data generated by Electrocardiograph (ECG) and Phonocardiogram (PCG). The present study deviates from these as it will extract features from auditory data instead. To classify the data correctly, three stages will be accomplished: feature extraction, feature selection, and classification. The goal is to determine which feature set can best classify the data using Gaussian Naïve Bayes and K-Nearest Neighbor classifiers. These methods are selected as they are commonly used in heartbeat classification studies. Three feature sets will be extracted: low-level, MFCC, and their combination. They will be combined with the techniques and evaluated using multi-class metrics. The pairing of feature set and method will be tested in classifying the heartbeat data into three main classes: normal, murmur, and having extra sounds.

Methodology

This study aimed to identify the appropriate feature set/s to be used in classifying heartbeat sounds into normal, murmur, noisy-normal, or having extra sounds. It was conducted in three stages, namely: feature extraction, feature selection, and classification. The process flow is shown in Figure 1.

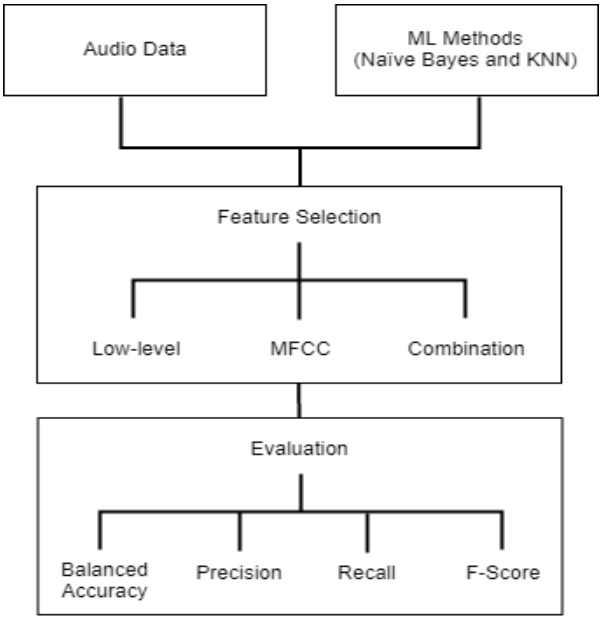

Figure 1. Process Flow

Data Gathering

The data used in this study are auditory data retrieved from a data competition website called Kaggle where thousands of data scientists exchange data and solutions for challenges. The data set was originally posted for a challenge in classifying heart sounds by Bentley, Nordehn, Coimbra, Mannor, and Getz (2011).

Two data sets were acquired. Data set A, which contains 136 audio files in WAV format, is acquired via the iStethoscope Pro IPhone App from the general public. Data set B, which contains 610 audio files, also in WAV format, is acquired using the digital stethoscope DigiScope from a clinic trial in hospitals. Both data sets include data for training and data for testing. Majority of the data is labeled, with 52 and 195 sound unlabeled files for set A and B, respectively, which can be used for other forms of testing and is not used for this study.

The organizations of Set A and Set B are found in Table 1 and Table 2. The numbers in the second and third columns represent the planned

Table 1. Auditory Data Set Organization for Set A

Class	No. of Data	Percentage	Training	Testing
Normal	31	36.9%	25	6
Murmur	34	40.5%	28	6
Extra	19	22.6%	16	13
Total	84	100%	69	15

Table 2. Auditory Data Set Organization for Set B

Class	No. of Data	Percentage	Training	Testing
Normal	200	46.3%	160	40
Murmur	66	13.3%	53	13
Extra	46	10.6%	37	9
Noisy-Normal	120	27.8%	96	24
Total	432	100%	346	86

partition of *a:b*, where *a* is the number of data used for training and *b* is the number of data for testing, following the 80/20 partition.

Feature Extraction

Feature extraction was performed on the data to come up with smaller space of variables, thus helping in analysis simplification. Three feature sets were used to extract the features for the classification of auditory data. These features were extracted using LibROSA, a library in Python that extracts features from auditory data (McFee et al., 2015).

(1) Feature Set 1

The features included in this set are low level features that can be directly determined from the raw audio input. These features are computed by slicing the audio into 20 to 40 ms frames.

(a) Spectral Bandwidth is the range where the sigma's spectral density is above zero or a given threshold value. The formula given by

$$SB = (\sum S[k](freq[k] - centroid^{p})^{\frac{1}{p}} \quad (1)$$

is abstracted in the function *librosa.feature.spectral_bandwidth(y)* where *y* is the audio time series (array). Other parameters include the audio sampling rate, spectrogram magnitude, window size, hop length, center frequencies for spectrogram bins, and power to raise the deviation from spectral centroid. This returns an array of the spectral bandwidth for each frame.

(b) Spectral Centroid is computed by calculating the "center of gravity" for each frame. In LibROSA, each frame is "normalized and treated as a distribution over frequency bins," where the mean or centroid is extracted for each frame. Formula in computing for the spectral centroid is given by the equation:

$$SpectralCentroid = \frac{\sum_{i=1}^{N} kF[k]}{\sum_{i=1}^{N} F[k]} \quad (2)$$

The function *librosa.feature.spectral_centroid(y)* is used, where *y* is the audio time series (array). Some other parameters for this function include the audio sampling rate of *y*, spectrogram magnitude, window size, hop length, and center frequencies. This function returns the center frequencies of each frame.

(c) Spectral Contrast is the difference in decibels between the high and low points of a spectrum. For every time frame, the spectral contrast is determined with the function *librosa.feature.spectral_contrast (y)*, where *y* is the audio time series. It also accepts other parameters such as the audio sampling rate of *y*, spectrogram magnitude, window size, hop length, frequency cutoff, the number of frequency bands, and quantile.

(d) Zero Crossing Rate is a measure of the rate of signal changes for each frame. This is determined by LibROSA with the function *librosa.feature.zero_crossing_rate(y)* where *y* is the audio time series (array). Several parameters are also accepted like the frame length, hop length, and the center of the frames. This function returns an array of the fraction of zero crossings in a given frame. The formula in computing for the zero crossing rate is given by:

$$ZCR = \sum_{n=-\infty}^{\infty} |sgn(s(n)) - sgn(s(n-1))| \quad (3)$$

(2) Feature Set 2

The second feature set is composed of the Mel-Frequency Cepstral Coefficients. These are coefficients widely used in audio analysis because they can represent the audio spectrum compactly. MFCCs are computed by splitting the signal into shorter frames. After that, the Fast Fourier Transform (FFT) is done to the frames and then mapped to the Mel scale. Using the Mel scale, the Discrete Cosine Transform (DCT) is then applied to each frame.

MFCC is extracted by LibROSA with the function *librosa.feature.mfcc(y)*, where *y* is the audio time series. It also accepts other parameters such as the sampling rate of y, the log-power Mel spectrogram, and number of MFCCs to return. It returns the MFCC sequence of the given audio sequence.

(3) Feature Set 3

The third feature set is the superset of the previously enumerated feature vectors with the following features: spectral bandwidth, spectral contrast, spectral centroid, zero crossing rate, and MFCCs.

Classification

After the features of the auditory data were extracted, the mean and standard deviation of the features for every frame were computed in order to have a single value per feature for every sound. This was used to train the input data. Two methods of classification were used. Like the feature extraction, these methods were programmed using Python 3, with the help of libraries NumPy and SciPy.

(1) Naïve Bayes Classifier

(a) Training
After the features were extracted for each of the training data, the mean of each feature for each class was computed, and the variance of the features for each class was also computed. These values served as the representative values of the means and variances of the features for each class, which were used in the classification of a test data.

(b) Classification
The representative values are plugged into the formula for the Gaussian Probability Density Function equation, along with the values of the features extracted from the test data. The Gaussian Naïve Bayes Classifier was used since the data has continuous values. The Gaussian distributions represent the likelihood of each class given a feature. The Gaussian Density Probability function given by:

$$X_i \sim N(\mu, \sigma^2) \quad (4)$$

The Gaussian Probability Density Function is defined by the equation:

$$N(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

The resulting value of the equation for each feature in a class is multiplied with each other, gaining one probability for

each class. The classification of a test data was based on the class with the highest computed probability.

(2) K- Nearest Neighbor

In classifying using this method, an input's feature was compared to the entire training data set to look for similarities. First, an arbitrary value of K was chosen, in this case, k = 10. Then, the Euclidean Distance of the test instance from the training instances was computed with the formula:

$$dist(a,b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \quad (6)$$

The distances were then sorted for optimization and the K-nearest neighbor was taken. A smaller value means it is nearer to the actual value. A simple majority was applied in getting the class of the input.

Evaluation

After the inputs were classified, the performance of the proposed algorithm was evaluated using multi-class performance features, that is, each input was classified to only one classification. The measures used are based on a study on the analysis of performance measures for classification tasks (Sokova & Lapalme, 2009), and are modified to fit the needs of the problem. The symbol N in the formula corresponds to the total number of data in the testing data set.

(1) Balanced Accuracy

This measures the average effectiveness of a classifier.

$$\frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \quad (7)$$

(2) Recall

This measures the effectiveness of the classifier in the correct prediction of the label of a data in a class.

$$\frac{tp}{tp+fn} \quad (8)$$

(3) Macro-Recall

This measures the macro average effectiveness of the classifier in the correct prediction of the label of a data in a class.

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i} \cdot (tp+fn)}{N} \quad (9)$$

(4) Precision

This measures the effectiveness of the classifier in class label identification.

$$\frac{tp}{tp+fp} \quad (10)$$

(5) Macro-Precision

This measures the macro average of the effectiveness of the classifier in class label identification.

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i} \cdot (tp+fp)}{N} \quad (11)$$

(6) F-Score

This measures the relations between the positive labels of the data and the classified labels.

$$2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (10)$$

(7) Macro F-Score

This measures the relations between the positive labels of the data and the classified labels based on a per-class average.

$$2 \cdot \frac{precision_M \cdot recall_M}{precision_M + recall_M} \quad (11)$$

A confusion matrix was used to tabulate the results that were used further for generating the performance measures. The matrix was extended for multiple classes, the horizontal labels are for the classified value, while the vertical labels are for the true value; the diagonals in the matrix show the true positive (TP) of the data. See Table 3 for reference.

Table 3. Multi-Class Confusion Metrics

	Normal	Murmur	Extra	Noisy-Normal
Normal	TP	-	-	-
Murmur	-	TP	-	-
Extra	-	-	TP	-
Noisy-Normal	-	-	-	TP

Results and Discussion

The classifier was mainly implemented and tested using the Gaussian Naïve Bayes Classification method in three runs: using the low-level features only, using the MFCCs only, and using the combination of the MFCC and low-level features. The data can be classified into four classes: normal, murmur, extra, and noisy-normal. This method was used for both data sets.

Table 4 shows the results from Set A (using the Gaussian Naïve Bayes Classification Method) having the balanced accuracy of 0.278, 0.389 and 0.333 for feature set 1, 2, and 3, respectively. Ideally, the f-score, precision, and recall should be equal or close to 1 since it signifies that the classifier was able to predict the data to their correct classes. Among those measures, the f-score punishes the extreme values from the precision and recall. The result of the performance measures as seen in Table 4 remains to be inconclusive because the precision and recall of the classes for each feature set contains extreme and inconsistent values, and the balanced accuracy and macro f-score are contradicting each other. One big factor for this is the data itself being inherently noisy, having been collected using the iStethoscope Pro IPhone App from the general public. Given that, the features extracted for each class in the training data were not distinct enough from each other.

KNN classification was not used in this test, as the amount of data was not enough. Since KNN is a lazy algorithm, it does more of the work in the testing phase and is mostly based on the instances in the training set. As the total training data for dataset A has a total of 69 instances, it cannot provide conclusive results given that nearest neighbor methods need a large number of training patterns for the classifier to provide good generalization (Nilsson, 1998).

Tables 5 and 6 show the evaluation summary for the results of the Set B data using Naïve Bayes and k-NN classifiers, respectively. Comparing the results between Table 4 and 5, Table 5 was able to get higher scores since the training data in data set B consists of 346 auditory data versus the 69 auditory data in data set A. In Table 5, the noisy-normal class produced mostly the highest precision and recall, followed by the normal class. These two classes have similar features. More specifically, both are fundamentally normal

Table 4. Evaluation Summary for Set A Dataset

	Precision			Recall			F-Score		
Class	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**
Normal	0.25	0	**0.33**	0.167	0	**0.5**	0.2	0	**0.4**
Murmur	**1**	0.5	**1**	0.33	**0.83**	0.167	0.5	**0.625**	0.286
Extra	0.11	**0.2**	**0.2**	**0.33**	**0.33**	0.2	0.167	**0.25**	**0.25**
Macro Averages	0.522	0.24	**0.573**	0.267	**0.4**	0.333	0.353	0.3	**0.423**

Table 5. Evaluation Summary for Set B Dataset using Naïve Bayes Classifier

	Precision			Recall			F-Score		
Class	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**
Normal	0.57	0.595	**0.677**	0.275	**0.625**	0.3	0.373	**0.61**	0.414
Murmur	0.227	**0.294**	0.25	**0.385**	**0.385**	**0.385**	0.286	**0.33**	0.303
Extra	**0.207**	0	0.179	**0.667**	0	0.556	**0.316**	0	0.203
Noisy-Normal	0.905	**0.933**	0.88	0.655	0.483	**0.759**	0.76	0.636	**0.814**

Table 6. Evaluation Summary for Set B Dataset using KNN

	Precision			Recall			F-Score		
Class	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**
Normal	**0.628**	0.609	0.556	0.675	0.7	**0.75**	**0.651**	**0.651**	0.638
Murmur	0.5	**0.667**	0.444	**0.538**	0.091	0.308	**0.519**	0.16	0.364
Extra	0	0	0	0	0	0	0	0	0
Noisy-Normal	**0.71**	0.548	0.6	0.759	**0.793**	0.517	**0.753**	0.48	0.556

heartbeats. Looking into the extra class, feature set 2 failed to classify the data because the extra class only comprised 10.6% of the training data.

In table 6, the noisy-normal class still showed the highest performance, and majority of the highest f-scores are classified using feature set 1. Feature set 2 and 3 produced extreme values for the precision and recall, thus affecting the f-score. KNN was not able to correctly capture any test data in the class extra, which Naive Bayes was able to do, except when using feature set 2. This is significantly affected by several factors: (1) the number of training data used for class extra (37 – training, 9 - testing); (2) KNN easily provides results less than or equal to zero when calculating for the distance, compared to Naïve Bayes where the probability is seldom resulting in zero; and (3) the feature set 2 does not provide enough features for the extra class to be uniquely recognized apart from the other classes.

After observing the performances of the methods using the three feature sets, the macro-averages were computed. Table 7 shows the macro average performance metric summary of the features for data set B. Naïve Bayes with feature set 3 dominantly showed the highest values, while it was feature set 1 for KNN.

As shown in Table 7, the highest scores for Naïve Bayes were produced by using feature set 3; for KNN, the highest scores were produced using feature set 1. It is significant that the balanced accuracy did not contradict with the f-score since the goal is to point out which feature set would produce the highest effectiveness while at the same time classifying the data to their correct classes. For this study, both precision and recall are given equal importance, that is why f-score was calculated.

Table 7. Macro-averages of the performance metrics for Data set B

Metric	**Method**	**Naive Bayes**			**KNN**		
	Feature Set	**1**	**2**	**3**	**1**	**2**	**3**
Balanced Accuracy		49.54%	37.31%	**49.97%**	**49.30%**	39.60%	39.37%
Recall		45.05%	**48.35%**	**48.35%**	**61.54%**	53.00%	53.85%
Precision		59.57%	60.11%	**62.69%**	**57.36%**	54.90%	49.89%
F-Score		51.31%	53.59%	**54.59%**	**59.38%**	53.93%	51.79%

Combining the standard low-level signal parameters with MFCC features helped in providing a more specific description of the class. Heartbeat signals are considered as non-stationary signals and since MFCC works best with non-stationary signals, MFCCs are a useful addition to the low-level features in classifying heartbeat. It also helped that the formula used is Gaussian, wherein the data points are assumed to be of normal distribution so outliers did not affect the result as much as KNN did.

For the case of KNN where k = 10, the highest averages were dominant in the test when feature set 1 was used. Feature set 1 has the least number of features since feature set 2 returns 20 MFCC values. This is where the curse of dimensionality occurs, as the points in high dimensional spaces have a tendency to disperse from each other than points in a low dimensional space (Dangeti, 2017). As KNN is a clustering algorithm dependent on the feature values of its neighbors, having less feature sets helps in preventing the curse of dimensionality. Increase in space also leads to overfitting.

Conclusion

The study was able to identify the best feature sets to use alongside Naïve Bayes and KNN for data set B. For data set A, the results are not enough to conclude on the best feature set to use since feature set 3 has the highest macro f-score while feature set 2 has the highest balanced accuracy, although the difference of the balanced accuracy for feature sets 2 and 3 is notably small – 0.056. For data set B, the feature set 3 (combination of low-level and MFCC features) can be best partnered with Naïve Bayes, while feature set 1 can be best partnered with KNN.

The selection of features strengthens the ability of these ML algorithms in classifying data according to their class. Feature selection provides an overview of the effectiveness of the ML algorithms, which proves how pre-processing in classification problems is vital.

To improve the results of this study, specifically the precision and recall in classifying the input data, spectral features (such as spectral flatness and spectral roll-off) may be included to help in the classification of the data. The data can also be pre-processed by looking for patterns and filtering the noise from the data to help in a more accurate classification. Furthermore, weights can be added to the features to see if there will be an increase in the precision and recall of each class.

References

Bentley, P., Nordehn, G., Coimbra, M., & Mannor, S. [n. d.]. (2011). The PASCAL Classifying Heart Sounds Challenge 2011 (CHSC2011) Results. Retrieved from http://www.peterjbentley.com/heartchallenge/index.html.

Dangeti, P. (2017). *Statistics for Machine Learning: Techniques for Exploring Supervised, Unsupervised, and Reinforcement Learning Models with Python and R.* Birmingham, UK: Packt Publishing.

Herrero, G., Gotchev, A., Christov, I., & Egiazarian, K. (2005). Feature extraction for heartbeat classification using independent component analysis and matching pursuits. In *Acoustics, Speech, and Signal Processing Proceedings (ICASSP'05). IEEE International Conference*, IEEE, 4, 725.

Kumar, D., Carvalho, P., Antunes, M., Paiva, R., & Henriques, J. (2010). Heart murmur classification with feature selection. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. IEEE, 4566–4569.

Lin, C., & Yang, C. (2014). Heartbeat classification using normalized RR intervals and morphological features. *Mathematical Problems in Engineering, 2014,* Article ID 712474, DOI http://dx.doi.org/10.1155/2014/712474.

McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., & Nieto, O. (2015). Librosa: audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, 18–25.

Mierswa, I. & Morik, K. (2005). Automatic feature extraction for classifying audio data. *Machine learning, 58*(2-3), 127–149.

Nilsson, N. (1998). *Introduction to Machine Learning*. Retrieved from https://ai.stanford.edu/~nilsson/MLBOOK.pdf

Singh, M., & Cheema, A. (2013). Heart sounds classification using feature extraction of phonocardiography signal. *International Journal of Computer Applications, 77*(4), 13-17.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437.