ECG Data Analysis & Classification

1. Introduction

Electrocardiogram (ECG) is a periodic signal which reflects the activity of the heart. It is used to investigate some types of abnormal heart function, for example, arrhythmias. One important method for the study of arrhythmias is the classification of heartbeats which is relevant to the analysis performed in the ECG data. The Heartbeat types include 5 classes which are N: normal, S: supraventricular, V: ventricular, F: fusion, and Q: unknown.

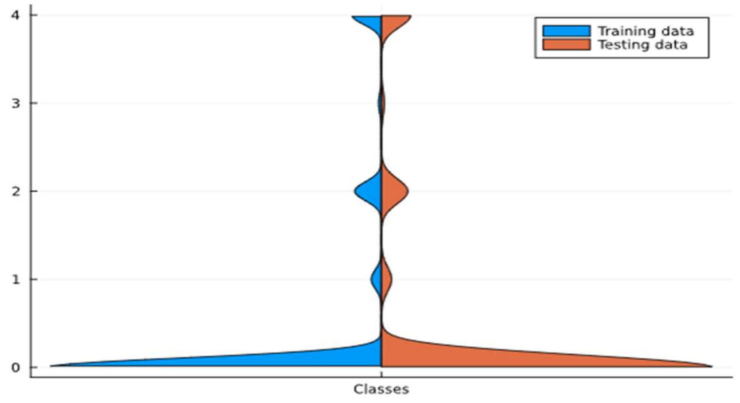
Three machine learning algorithms KNN, Decision Tree, and SVM are used in this assignment to do the classification of the ECG data. This report contains the methods used including exploratory data analysis and classifiers, experiment results, and discussion and conclusion of the three algorithms. The purpose of this work is to compare the classification process and classification accuracy between the three algorithms, find and justify the best classifier to use in the ECG data.

2. Method

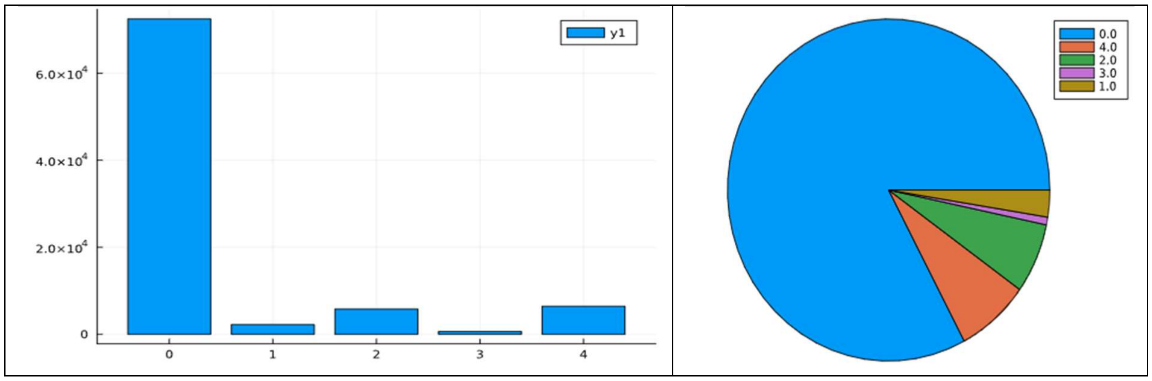
2.1. Exploratory data analysis

Exploratory data analysis is implemented to find out if there any interesting pattern of the data, the following methods are used:

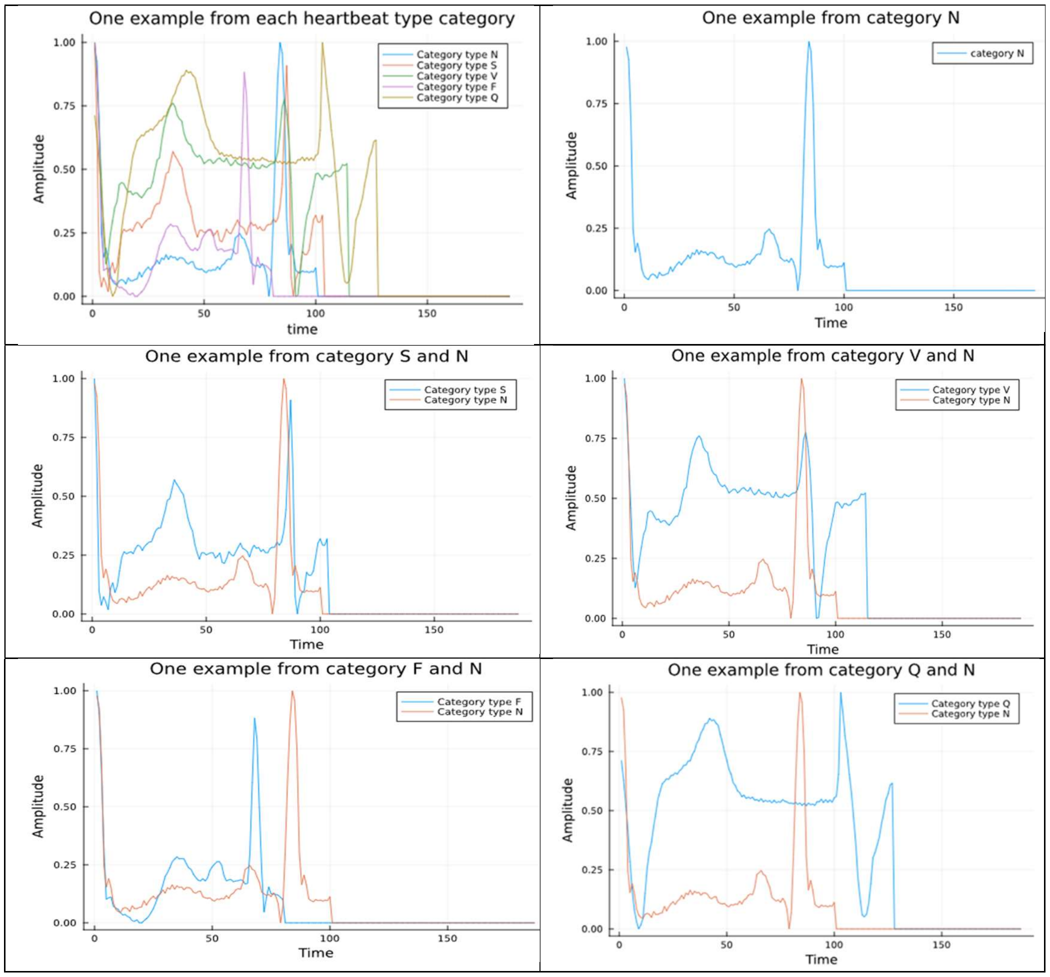
- Carry out statistical comparison of the training dataset and testing dataset. Find that at least 95% confidence that both datasets are statistically identical.



- Check the number and the proportion of each classification in the training dataset, find the dataset is highly imbalanced. I mentioned to fix the imbalanced data issue in section 2.2.



- Plot and compare a sample of each heartbeat type, find that for different heartbeat types, the length, the timing and the amplitude of different waves are different, so the classification type is highly likely to be dependent on these factors. Therefore, the length, the timing and the amplitude of different waves can be used as features when we are doing heartbeat type classification if applicable.



- Clustering analysis: use kmeans to do clustering of the training data, cluster the training data into 5 groups as the training data has 5 heartbeat types. Notice the total number in each heartbeat type in the training data is not the same as the total number in each cluster group. This implies the classification is not totally dependent on the distance between samples as Kmeans does. But worth to use clustering group number as a feature if applicable when doing classification.

2.2. Make the training dataset balanced

From the exploratory data analysis, I noticed the training dataset is imbalanced. AUC is good at handling imbalanced data, but AUC is for binary classification. Since I have 5 classifications in this task, I cannot use AUC directly. Therefore, I used Accuracy to evaluate the classifier. To use Accuracy, the prerequisite is that the training dataset needs to be balanced, because if we use an imbalanced dataset, the classification accuracy is not reliable. For example, if the training dataset is imbalanced with 80%

data which belongs to the heartbeat type “N”, a naive classifier build from the training data can easily achieve 80% accuracy rate by just simply predicting all the ECG data are the heartbeat type “N”.

I used function “MLUtils.getobs” and “MLUtils.numobs” to make the training dataset balanced.

2.3. Feature extraction

From the exploratory data analysis, find that the different heartbeat type classifications are driven by different wave intervals, amplitude, and timing. I tried to extract the RR interval and the QRS interval by using several ways, but the calculation to extract RR interval and QRS interval features can be very complex. In the end, I decide to use below simple statistical calculations as the features:

- Cluster group number
- Standard deviation
- Mean
- Max value (may represent one R wave amplitude)
- Max value index/column (may represent one R wave timing)

2.4. Classifier models

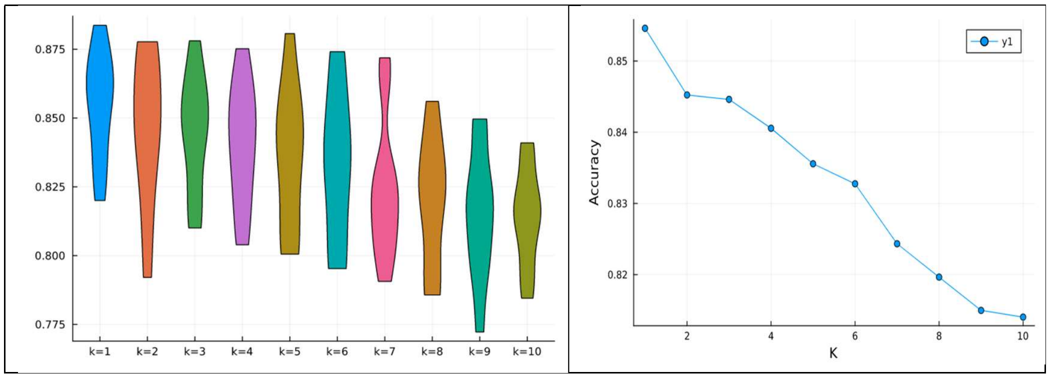
KNN, Decision tree and SVM

2.4.1. KNN

KNN is to find the k nearest neighbours in the training data that close to the each of the test data by using distance calculation, then classified the test data to the classification which has the majority votes among all the k nearest neighbours. Uniform method will make all the k nearest neighbours have the same weight, does not matter they are close or far away. Inverse distance method will make the neighbours that are closer have more weight.

The following steps are used to produce the classification of the test data:

- Set the distance calculation and method to be default, which are Euclidean and uniform. Only tune the model based on the K.
- Performance tuning to find the best hyperparameters: use “evaluate!” and 10 folds cross validation on the training data to evaluate different models for k from 1 to 10, get the accuracy information.
- Create a violin plot and a line plot find when k=1, the accuracy is the highest, therefore select k=1 to build a KNN classifier to do the classification for the test data.



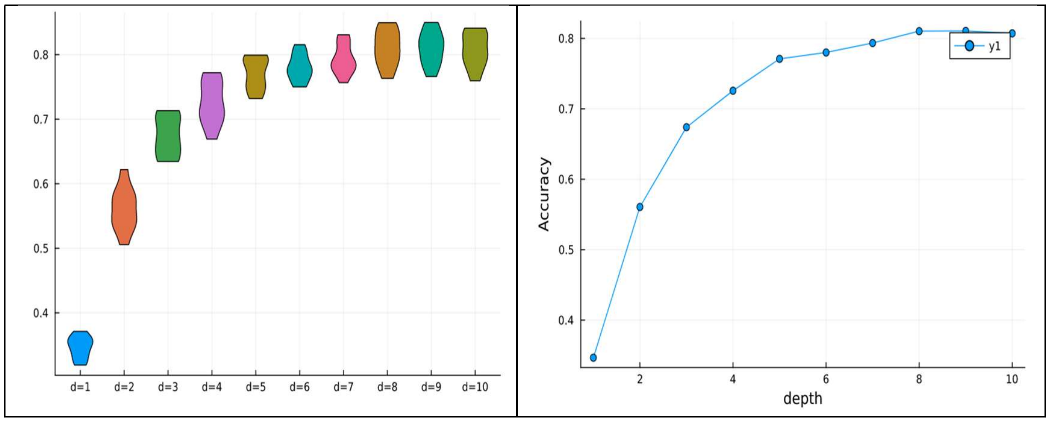
- Use “predict_mode” to get the classification of the test data.

2.4.2. Decision Tree

Decision tree is to find the best split of features and its values that deviate least from the mean of the response variable at a certain point.

The following steps are used to produce the classification of the test data:

- Performance tuning to find the best hyperparameters: use “evaluate!” and 10 folds cross validation on the training data to evaluate different models for depth from 1 to 10, get the accuracy information.
- Create a violin plot and a line plot find when depth=8, the accuracy is the highest, therefore select depth=8 to build a decision tree classifier to do the classification for the test data.



- Use “predict_mode” to get the classification of the test data.

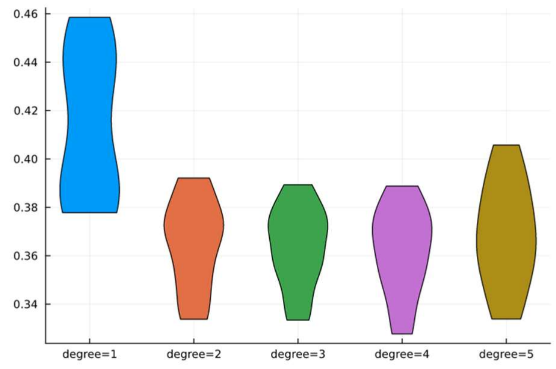
2.4.3. SVM

SVM focus on selected marginal data points (“support vectors”) and attempt to locate an optimal classification hyperplane with maximal margins between the classes.

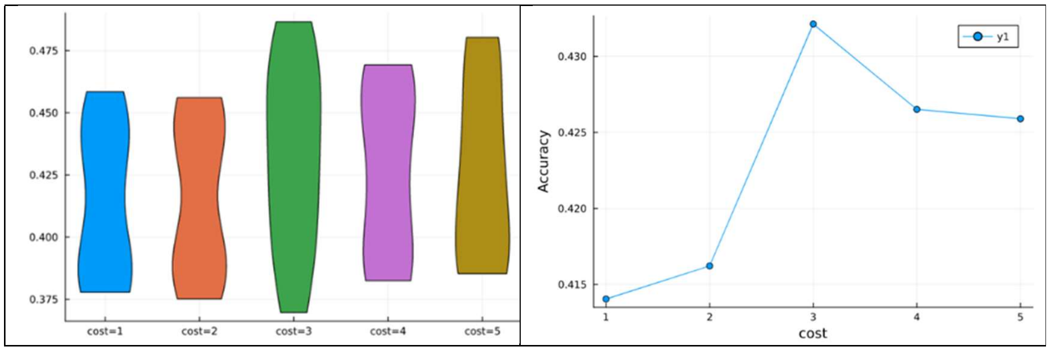
Traditionally SVM is for binary classification. For multiclassification problem, SVM needs to use one to one approach, or one to rest approach to build multiple SVM classifiers. For any sample, the classifier has the highest predicted classification probability wins, and the sample will be the classification with the highest probability. But thanks to Julia SVC (C-Support Vector Classification) function, SVM in Julia can directly do multiclassification by using one to one approach.

The following steps are used to produce the classification of the test data:

- Feature extraction: SVM was slow with all the features, discussed in section 2.3.
- Feature evaluation and selection: evaluate the features extracted, select the sub features have the highest accuracy of classification, discussed in section 3.1.
- Performance tuning to find the best hyperparameter degree: use “evaluate!” and 10 folds cross validation on the training data to evaluate different models for degree from 1 to 5, get the accuracy information.
- Create a violin plot find when degree=1, the accuracy is the highest, therefore I select degree=1.



- Performance tuning to find the best hyperparameter cost: use "evaluate!" and 10 folds cross validation on the training data to evaluate different models for degree=1, cost from 1 to 5, get the accuracy information. Only trying for cost from 1 to 5, because SVM algorithm is slow.
- Create a violin plot and a line plot find when cost=3, the accuracy is the highest. Therefore, select degree=1 and cost=3 to build a SVM classifier to do the classification for the test data.



- Use "predict_mode" to get the classification of the test data.

3. Experiment Results

3.1. Feature evaluation and feature selection

After obtaining the features from the feature extraction process, I did the feature evaluation process. I tried a few combinations of different features that I have extracted and use the SVM classifier degree=1 to evaluate the accuracy on the 10 folds cross validation of the training data. I selected the subset of features (including cluster number, standard deviation, mean) which has the highest accuracy rate as my final dataset to do the classification for SVM.

3.2. Performance evaluation

I computed the accuracy and confusion matrix of each classifier, using a table to compare the accuracy, confusion matrix, speed, whether needs feature extraction on the three classification algorithms.

- Accuracy: the accuracy percentage of classifications get correctly classified by using each classifier, from the whole testing dataset perspective.
- Confusion matrix: predict classification result in each heartbeat type.
- Speed: the computation speed of each classifier.
- Needs feature extraction or not: whether the classifier needs feature extraction and selection process before doing the classification.

	Accuracy	Confusion matrix	Speed	Needs feature extraction?
KNN	0.7795	Majority were classified correctly for each classification	Quickest	no
Decision Tree	0.7508	Majority were classified correctly for each classification	Quick, but a little bit slow than KNN	no
SVM	0.4813	Only Class 3 majority were classified correctly, class 0, 1, 2, 4 were very poorly classified	Quick with subset of features, Very slow with all the features	Yes, the feature extraction and selection part are both time consuming

4. Discussion & Conclusion

Below is my discussion regarding each algorithm used based on the experiment results, and my choice of the classifier for this ECG data.

4.1. KNN

- Among the 3 algorithms, KNN has highest accuracy rate and quickest speed.
- In KNN classifier, majority samples were classified correctly for each classification.
- KNN does not need feature extraction, the raw data can be used directly for classification, the data preparation time is short.
- KNN algorithm is quick that is because the training data in KNN does not need to be trained to build a model, the training data is only used to do the classification of the test data. There is no training time for training data, therefore KNN is quick.
- KNN is sensitive with noisy data. But in this task, as the data have already been normalized to be between 0 and 1, the previous noisy data is no longer that noisy in the normalized dataset. So KNN performed good.

4.2. Decision tree

- Decision tree has the 2nd best accuracy rate and is relevantly quick too.
- In Decision tree classifier, majority samples were classified correctly for each classification.
- Decision tree does not need feature extraction in this case, the raw data can be used directly for classification, therefore the data preparation time is short.
- Decision Tree is slower than KNN because decision tree needs to train the training data to build a model, then used the trained model to predict the test data. Whereas KNN does not

need to train the training data, the training data is only used when there is test data need to do classification.

- In the process of building the model, it can easily cause problem of overfitting, so choose the correct depth of the tree is very important.
- Decision tree is robust for handling outliers by doing the right split. But since the data is already in between 0 to 1, the outliers are probably no longer that noisy. So, this advantage of decision tree cannot be shown in this case.

4.3. SVM

- SVM has a very low accuracy rate, based on the features I have extracted.
- The feature extraction and selection process were very time consuming, and from my experiment I can see that extracting the relevant and right features are super important, otherwise it will lead to incorrect classification.
- In SVM classifier, only Class 3 majority samples were classified correctly, class 0, 1, 2,4 are very poorly classified. Errors occur between all the classes.

4.4. My choice of classifier

After the comparisons, I prefer KNN algorithm for this case, because:

- KNN has the highest accuracy rate.
- KNN does not need much data preparation as KNN does not need feature extraction and the data is already normalized.
- KNN is very quick to do the classification as KNN does not need to train the training data.
- KNN is not robust for noise data, but since all the data are normalized in between 0 to 1 in this case, KNN probably is not that much affected by noisy data as noisy data is no longer that noisy.

4.5. Conclusion

I have used KNN, Decision tree and SVM to do classification of the ECG data. This is a very interesting process to see how each algorithm react on the ECG data.

The future work for the ECG data classification perhaps try to use ROC and AUC to do the performance evaluation instead of Accuracy. As AUC is for binary classification problem, so I cannot use AUC directly. My idea is to have different binary combinations of the data, do the binary classification, produce the AUC, and then combine all the AUCs together.

More detailed experiment contents, please refer to the Jupyter notebook.

Reference:

[1] INFO411 lecture slides

[2] INFO411 labs

[3] Luz, E. J., Schwartz, W. R., Cámara-Chávez, G., et al. (2016). ECG-based heartbeat classification for arrhythmia detection: A survey. Computer methods and programs in biomedicine, 127, 144– 164. http://www.sciencedirect.com/science/article/pii/S0169260715003314

[4] Mondéjar-Guerra, V., Novo, J., Rouco, J. et al. (2019). "Heartbeat classification fusing temporal and morphological information of ECGs via ensemble of classifiers. Biomedical Signal Processing and Control. 47: 41–48. http://www.sciencedirect.com/science/article/pii/S1746809418301976

[5] API Reference https://juliaml.github.io/MLUtils.jl/dev/api/

[6] Evaluating Model Performance

https://docs.juliahub.com/MLJ/rAU56/0.15.2/evaluating_model_performance/

[7] Join an array of strings into a single string in Julia – join() Method
https://www.geeksforgeeks.org/join-an-array-of-strings-into-a-single-string-in-julia-join-method/

[8] Constructors and Types https://docs.julialang.org/en/v1/base/arrays/

[9] Svm.jl https://docs.juliahub.com/PosDefManifoldML/DEM8i/0.4.2/svm/