Mohammadreza Hajy Heydary

Analysis and Classification of Four Categories of Arrhythmia

1. Introduction

Arrhythmia refers to an abnormality in the ordinary pattern of heart impulses. This abnormality, for instance, could be in the form of slow or rapid heart impulses. Arrhythmia is known to be one of the most common heart conditions with more than three million occurrences per year in the United States. If left undetected or untreated, an arrhythmia may result in severe or even fatal conditions such as myocardial infarction. A standard method to diagnose an arrhythmia is to use Electrocardiogram (ECG) test records. Formerly, performing an ECG was only possible using specialized machines and under the supervision of experts. However, modern wearable gears are now equipped with sensors capable of recording ECGs. The rapid proliferation of gears equipped with ECG sensors demands the employment of advanced analytical tools to properly utilize the recorded datasets.

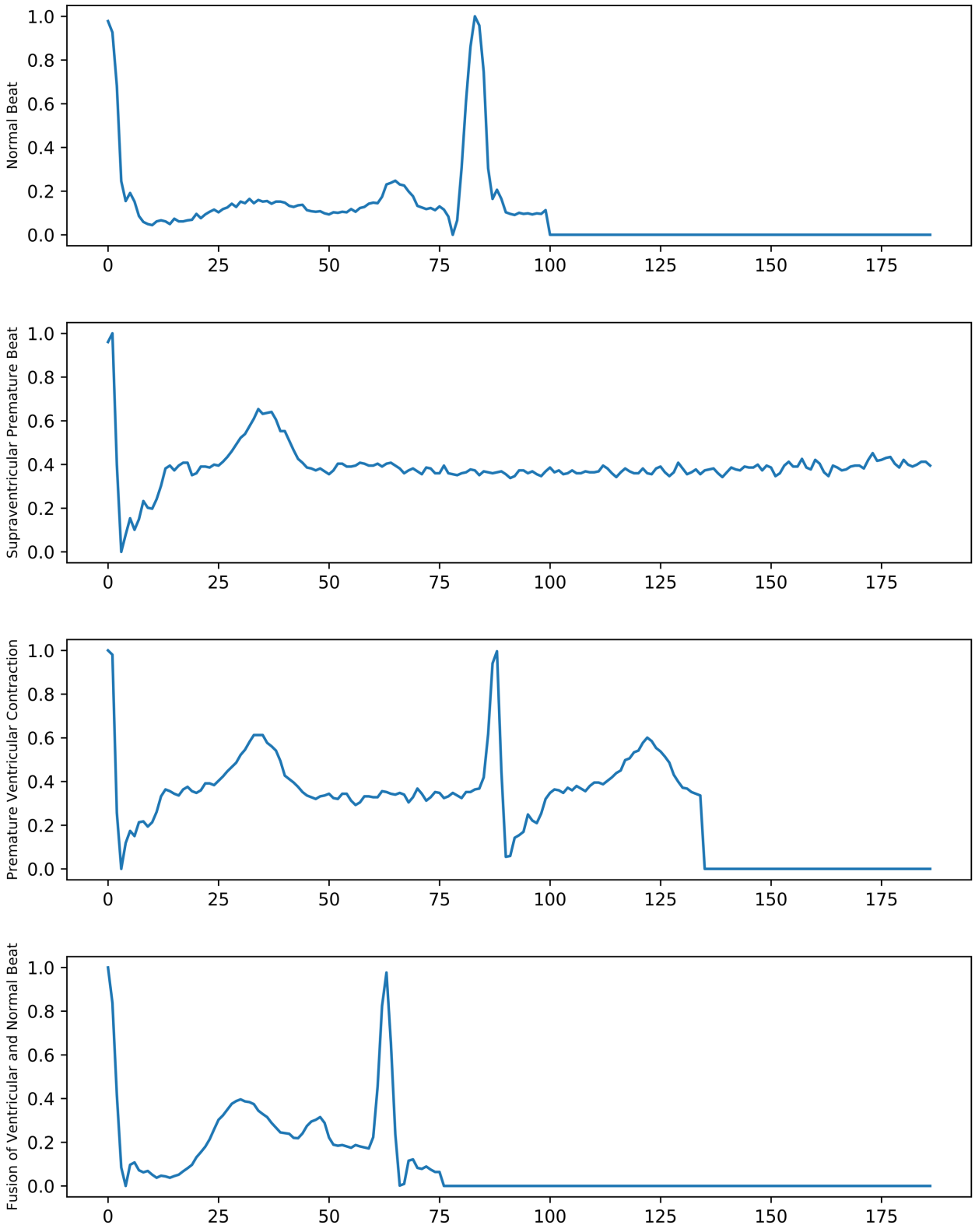
In this project, we are interested in examining the ECG records and developing a classifier capable of detecting arrhythmia leveraging the *MIT-BIH Arrhythmia Dataset* and the *PTB Diagnostic ECG Database* [1]. These datasets consist of more than one hundred thousand samples labeled into four different classes of arrhythmia. The feature space consists of 187 dimensions each recording the magnitude of the recorded electric pulse at 125 Hz sampling frequency. Study of this problem falls under the umbrella of time series classification which has been an object of intense inquiry. The high dimensionality of the feature space aligned with the relatively small, ~110k, number of samples proposes significant challenges both in the preprocessing and classification steps. Nonetheless, we believe it would be feasible to classify the data with a reasonable accuracy.

2. Dataset Characteristics and Preprocessing

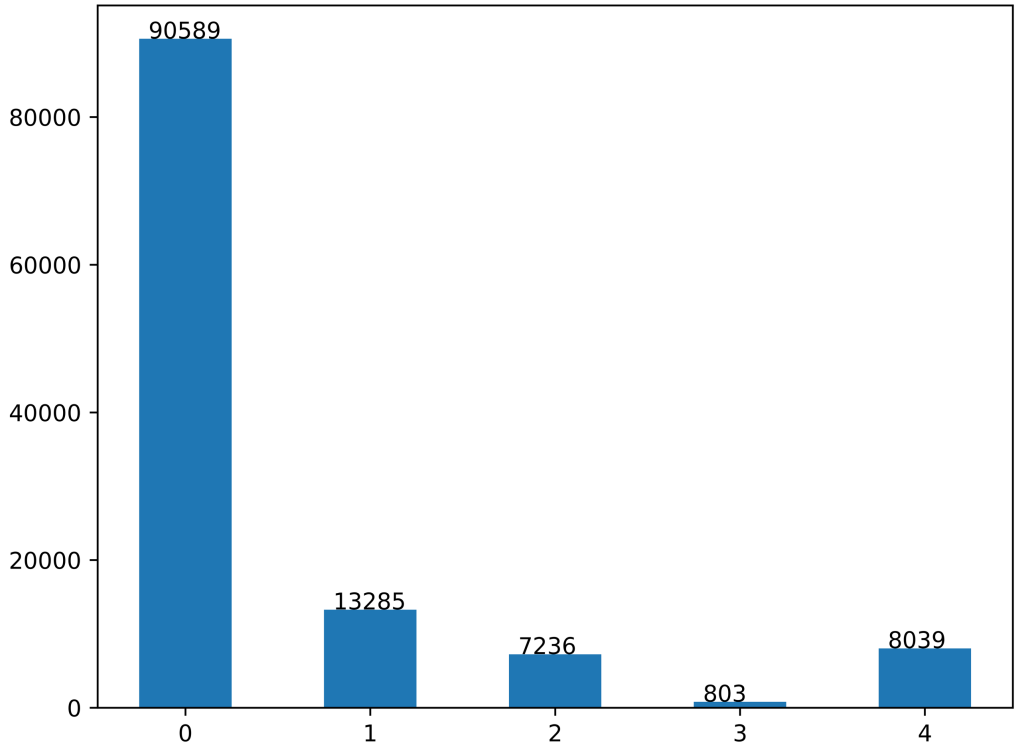
The datasets that we use in this study are open source and available on *Kaggle's* website. There are a few key features to this dataset that makes the preprocessing step easier. First, the dataset has been already cropped, down-sampled, and padded with zeroes if necessary. Thus, there is no need to be concerned about missing values. Further, the dataset is partitioned into train and test subsets. However, we will ignore the existing partitioning due to the skewness of the data. The last column of the dataset contains the labels denoting the type of arrhythmia with the following mapping convention:

Class	Encoded Label
Normal beat	0
Supraventricular premature beat	1
Premature ventricular contraction	2
Fusion of ventricular and normal beat	3
Unclassifiable beat	4

Following is a visual representation of each type of beat mentioned previously:

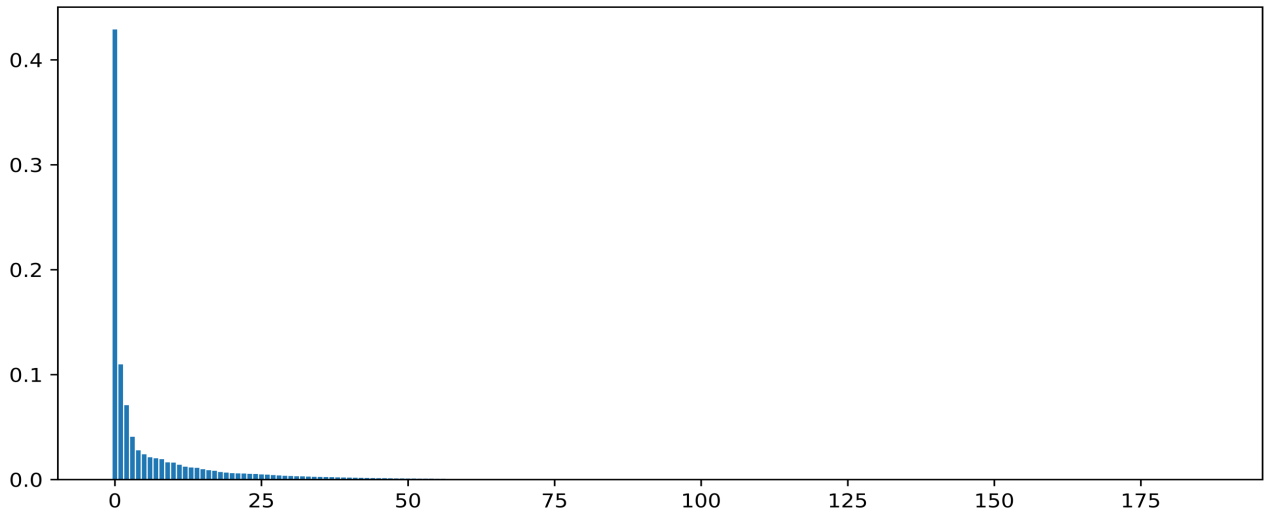


A major area of concern in classification studies is the distribution of samples in each class and whether the distribution is representative of the real world. The datasets under investigation suffer from a skewed distribution. To alleviate the issue, we augmented both datasets into one and ignored the existing train test partitioning. The resulting distribution of the data is plotted in the following bar graph.

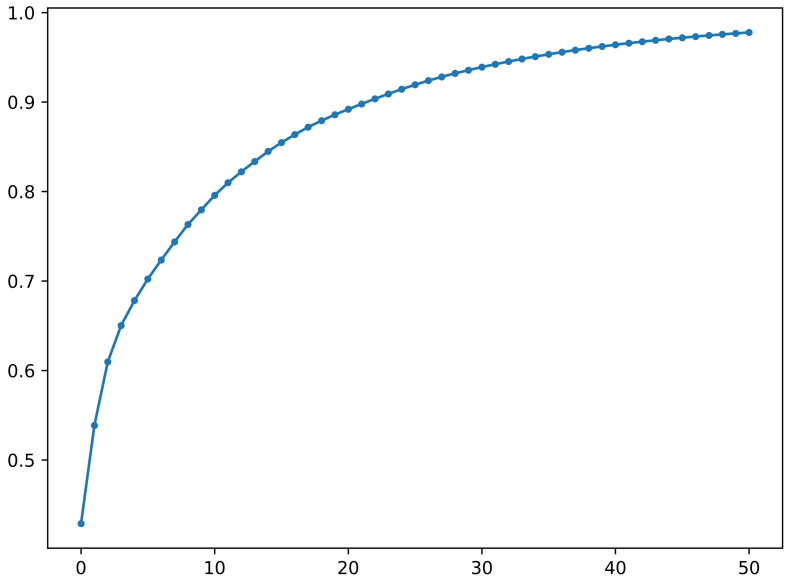


Looking at the bar plot, we can see a very skewed distribution. Generally, in the case of having such datasets, we should match the number of samples in each class to a relatively close neighborhood. In other words, more data need to be collected. However, collecting new data is not always feasible as it is the case with this study. Hence, to make the study valid, we assume that the distribution of data follows the general trend in the frequency of observing a particular class of arrhythmia in an unbiased sampling of a population and there is no bias in the dataset. Further, we shall use K-fold to minimize the impact of this anomaly on the results that we report in this analysis. The main concern, therefore, in the preprocessing step is now to deal with the high dimension of the dataset compared to its size.

One approach to extract features from the raw data and to project it to a lower dimension is to employ Principle Component Analysis (PCA). If we perform the PCA decomposition on the dataset and plot the variance explained by each feature, we get the following bar plot.



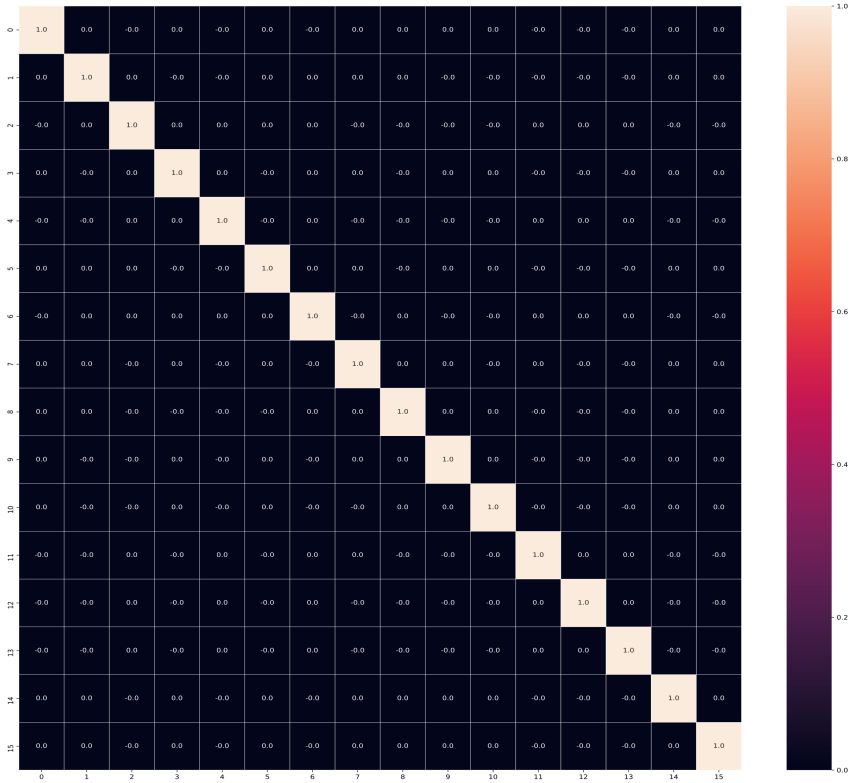
Looking at the plot, we can observe that there is a large number of features that do not explain much of the variance in the dataset. Approximately 50 features out of 187 of the original feature-space explain the majority of the variance in the data. Thus, we can initially eliminate about 130 features from the dataset which is a significant reduction. However, picking all the fifty-dimensions results in a space with still a high dimensionality. Indeed, pushing the limits and trying to pick all the significant features can cause over-fitting and increase the model complexity. Thus, we add the features from the most important to the least important for the first 50 features and plot how much of the variance is explained by adding features one by one.



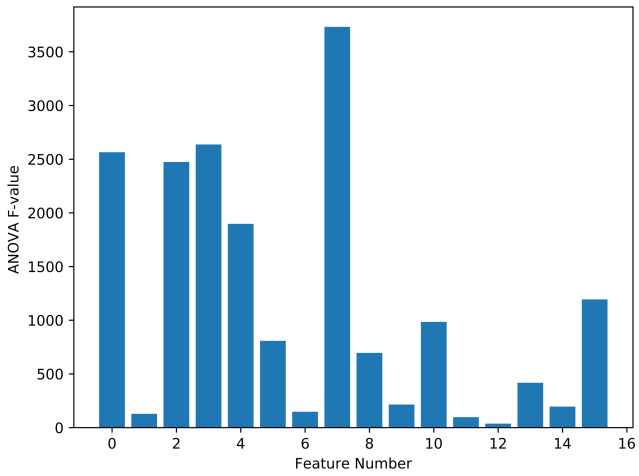
Looking at the figure above, we can observe that as we increase the number of dimensions, the change becomes less and less significant. Thus, to avoid overfitting and unnecessarily

complex model, we shall only use the first 16 features which account for approximately 85% of the variability in the dataset.

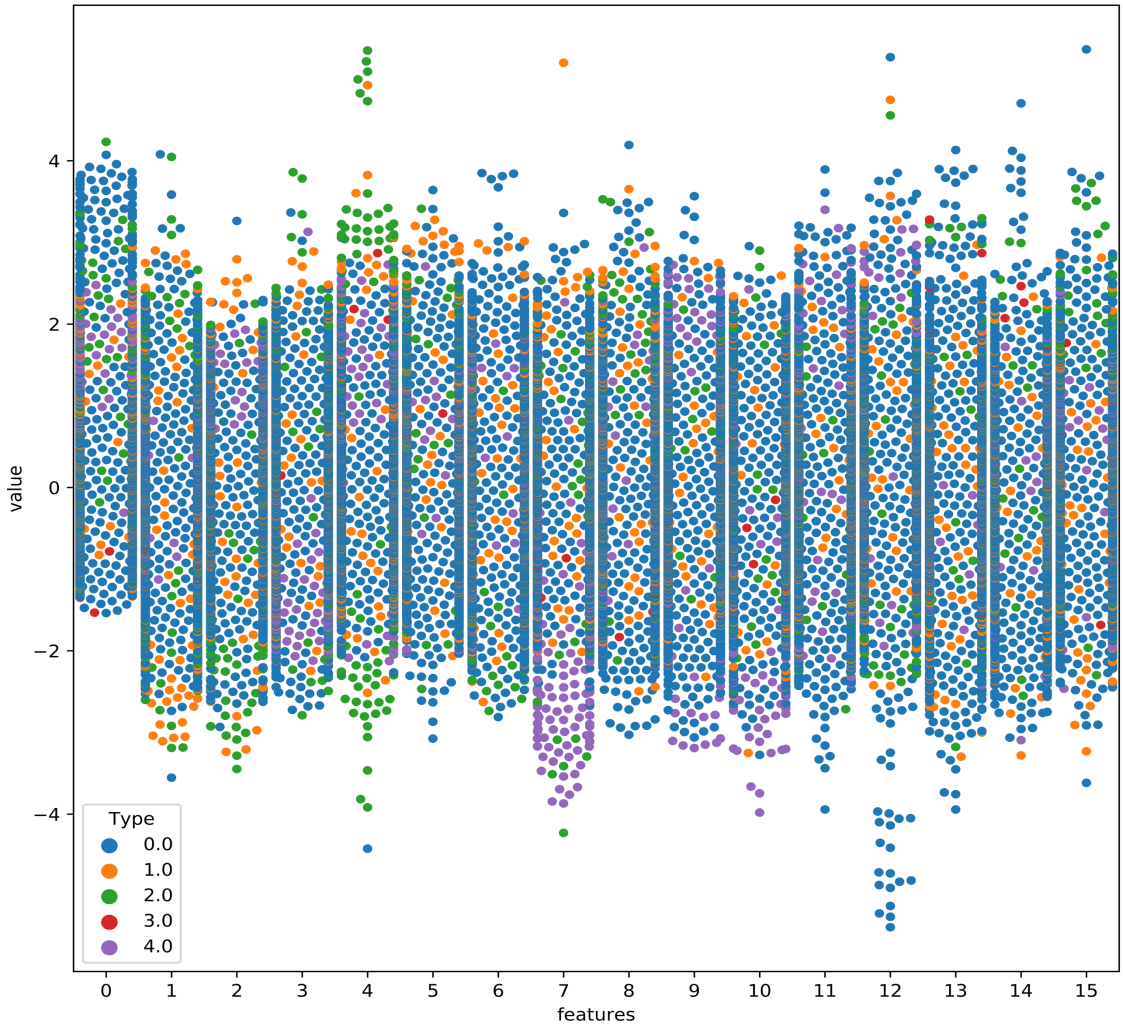
So far, we have projected the data into space with a much smaller number of dimensions. Now, we shall examine the contribution of each of these features to the final classification task. This could potentially assist us to eliminate more features that are redundant and have a negative impact on the skill of the model. We do know that PCA maps the data into a form with zero correlation between features. This could be observed experimentally via a correlation heat map of the features. The heat map shows zero correlation (indicated in black) between the features. Therefore, we shall not be concerned about having features that are highly correlated and, hence, redundant.



Next, we shall use the scikit-learn *Select K Best* method using ANOVA F-value between labels and features to find the score of each feature and their contribution to the final classification. To this end, we shall divide the data into train and test subsets using the K-fold method. K-fold is a requirement for this study as the data distribution is skewed and different combinations of train and test should be considered. We use ten folds in total, nine of which are used to train the model and one is used for testing purposes. This partitioning shall be used through the study from this point unless specified otherwise. We run the K-best method on the training partitions and find the means of scores reported by the algorithm. We let k, the number of features to be selected, to be 10. As the bar-plot also indicates, there are features that do not have a major contribution to the classification task and shall be dropped.



We can also visualize the distribution of values of each class in the features via a swarm plot. However, the relatively large size of the dataset does not permit us to plot the entire dataset. Thus, we take a random sample of the data plot those. This should give a decent visualization of the data. Following is a swarm plot of a 3% random sample of the original dataset:



Even though this plot only includes 3% of the original data, we can observe boundaries of classification for the features that had a high-score from the K-best method such as feature number 7.

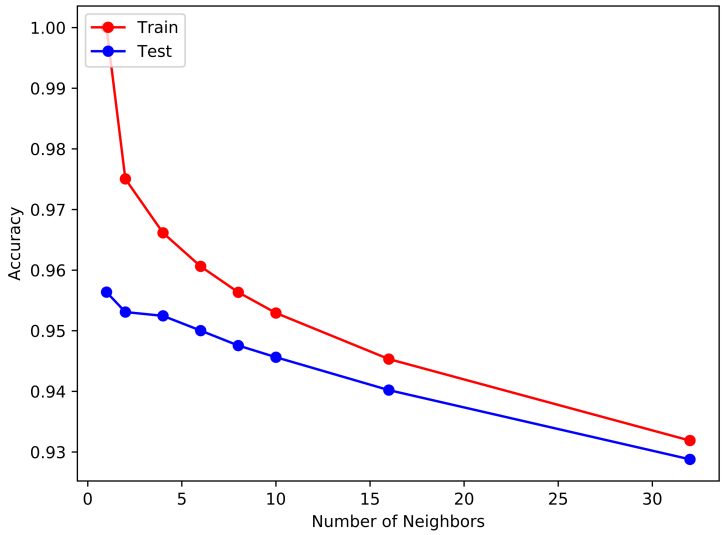
3. Model Development

Considering that the dataset at this point only contains ten features that explain the majority of the variance in the data and have the highest estimated contribution to the classification on average, we shall move to the training phase. In this study, we will compare and tune four different approaches to generate a model: KNN, SVM, Decision Tree, and Random Forest Classifier. The model that yields the highest accuracy score on average shall be considered as the final model that we propose for future studies.

3.1. KNN

The first model that we consider is KNN. We used the K-fold partitioning schema that was described previously to train and test the model and took the average over all the splits to develop an understanding of the performance of the model for each set of hyperparameters. We also tune the hyperparameters of the model, in this case, the number of neighbors. The list of combinations that we test the KNN model on is: [1, 2, 4, 6, 8, 10, 16, 32]. We used the default metric, *Euclidian distance,* to measure the distance between the points in the space.

N	Mean Train Accuracy	Mean Test Accuracy
1	1.0	0.9563658755619517
2	0.9750372362175312	0.9530812937886207
4	0.9661401596469077	0.9524560193620049
6	0.9606101698033974	0.9500300821513369
8	0.9563390181684281	0.9475624317132759
10	0.9529302468486212	0.9456366730514315
16	0.9453299845626834	0.9402094824771542
32	0.9318912736802174	0. 9287964866149414



Looking at the accuracy table and its respective plot, we can see the highest overall accuracy is attained when the number of neighbors is set to one. However, choice of N=1 possesses the potential to overfit and not resulting in a decent prediction once tested on unknown cases. Therefore, we will pick N=4 for the number of neighbors as the sacrifice in accuracy is negligible and the chance of overfitting is smaller.

3.2. Support Vector Machine

The next model that will consider for the purpose of classifying this dataset is SVM. The main hyperparameters of this model are: kernel type and margin (C). We test the model on four different kernels: ['linear', 'poly', 'rbf'] and four different values of C: [0.0001, 1, 2, 4]. The following table summarizes the results of the model on both test and train sets:

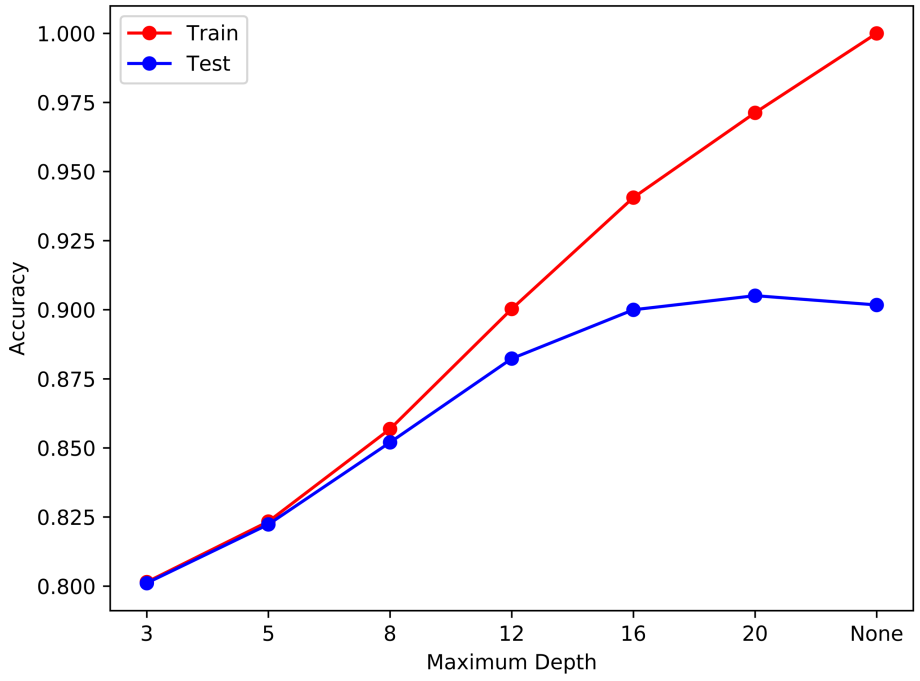
Kernel	C	Mean Train Accuracy	Mean Test Accuracy
Linear	0.0001	0.755215975389973	0.7552187641306364
	1	0.8071950997068844	0.807147851818389
	2	0.8072015837356634	0.8071728608496983
	4	0.8071960260359333	0.8071728608496983
Poly (deg 3)	0.0001	0.7552104176387614	0.7552104287134488
	1	0.8495398157079768	0.8490729655385237
	2	0.8544834604734595	0.8541333233835227
	4	0.8587601717644404	0.8582516567673915
RBF	0.0001	0.7552104176387614	0.7552104287134488
	1	0.8761893661983393	0.8756586425212829
	2	0.8830365506982256	0.8822112978803165
	4	0.8886693589735056	0.8876051015933785

One important point that worth mentioning is that SVM was much slower than other methods that we used. Further, if we look at the train and test accuracy results, there is a clear choice for the hyperparameters and that is a Radial Basis Function (RBS) kernel (aka squared-exponential kernel) and a soft margin with a magnitude of two. Even though choosing a margin of size 4 increases the accuracy slightly, the decrease in the rate of improvement of the model is an indication of overfitting.

3.3. Decision Tree

The next method that we use to generate a model is a Decision Tree. The main hyperparameter that needs to be tuned in this model is the maximum depth of the tree that is being constructed. We let the criterion function to remain to its original value, GINI. Change of criterion is unlikely to have a noticeable impact on the performance of the model. The set of depth that we test the model on is: [3, 5, 8, 12, 18, None] where None indicates that there is no restriction on how deep the tree can grow. The following table indicates the accuracy score measures for each combination.

Max Depth	Mean Train Accuracy	Mean Test Accuracy
3	0.8014502089972098	0.8010788410867749
5	0.8233524835844859	0.8223040400384655
8	0.8568334752195312	0.8519908605295189
12	0.9002684410686758	0.8823028352322817
16	0.9405725210049823	0.8999682279323332
20	0.9712514601030527	0.9050785990040504
None	1.0	0.9017188629438355

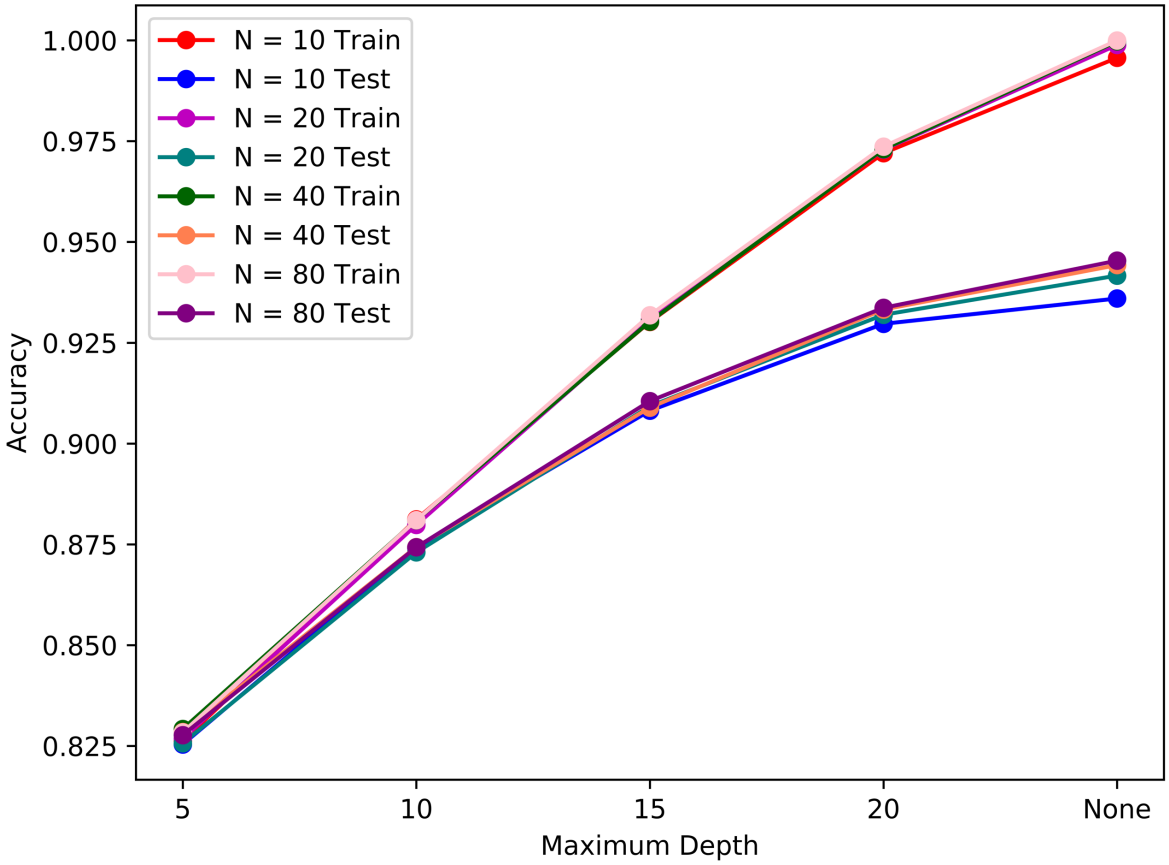


Looking at the plot, we can see that as we increase the maximum depth of the tree, the accuracy score increases as well. However, if we do not impose any restriction on how big the tree can grow, the training accuracy increases while the testing accuracy decreases. In other words, we will be having an overfitting issue. Thus, the optimal choice of the hyperparameter is 20. Moreover, the decision tree approach is a lot faster than the SVM approach.

3.4. Random Forest Classifier

An alternative that can be used to boost the performance of the generated models is to use an ensemble approach. In particular, we use a Random Forest Classifier and compare the results of this method to other models. Two main hyperparameters that need to be tuned in this method are the number of trees in the forest and the maximum depth of each decision tree. We vary the value of the number of trees in the forest and maximum depth in the following order respectively: [10, 20, 40, 80] * [5, 10, 15, 20, None] where None indicates that there is no restriction implied on how deep any given tree can grow. The results of the classifier are as follows:

Number of Trees	Maximum Depth	Mean Train Accuracy	Mean Test Accuracy
10	5	0.826234206331838	0.8253716938851093
	10	0.8811960179846954	0.8741662351798549
	15	0.930185978160732	0.9081548731830242
	20	0.972033251404213	0.9296886292336612
	None	0.9956686368910358	0.9359745366580319
20	5	0.8268260873753398	0.8256720406137319
	10	0.879821395766619	0.8730073418359483
	15	0.9306500359126509	0.9092721026702556
	20	0.9728956278752664	0.9319396373571134
	None	0.9988606552700491	0.941660143220323
40	5	0.8291872184390812	0.8284397751114779
	10	0.8809727504609656	0.8743415082826826
	15	0.9302628391558032	0.9089220610128578
	20	0.9730160630330762	0.9332484445083775
	None	0.9997850990995593	0.9442611832672514
80	5	0.8283480045271521	0.827672725606682
	10	0.8809931451789111	0.874274715846943
	15	0.9318273641831392	0.9105309445471959
	20	0.9736422299124697	0.9336653668660843
	None	0.9999814739853035	0.9453616994957995



Looking at the table and the plot above, we can see that the overall accuracy of the model increases as the as the number of trees and the maximum depth increase. This pattern continues all the way to having no-limit on how deep a tree can grow and a maximum number of 80 trees in the forest. In such a case, we would pick the model with the highest number of trees. The more tree that we have in the forest, the better the performance of the model is as the variance and bias due to sampling would be minimized. Nonetheless, having more trees in the forest comes at the cost of having a computationally expensive training phase and a slightly slower prediction rate. Thus, we should not assume that increasing the number of trees does not have a downside. Moreover, increasing the maximum length indicates a pattern of increase in the accuracy of both train and test data. However, if we do not limit the maximum depth, there is a good chance of model overfitting as the training accuracy becomes dangerously close to a perfect score. Hence, we pick a maximum depth of 20 in our final model.

4. Conclusion

To summarize, we tested and tuned four different models and measured their accuracy using the K-fold method with ten folds. The following table summarizes the best accuracy achieved in each model.

Model	Mean Train Accuracy	Mean Test Accuracy
KNN	0.9661401596469077	0.9524560193620049
SVM	0.8830365506982256	0.8822112978803165
Decision Tree	0.9712514601030527	0.9050785990040504
Random Forest	0.9736422299124697	0.9336653668660843

Numerically, looking at the results, we can see that the Random Forest method has the highest accuracy on the training dataset whereas KNN has the highest accuracy on the test dataset. Moreover, we can see that the performance of the Decision Tree and Random Forest are very similar on the training set. However, the ensemble method (RFC) surpasses the normal decision tree on the test dataset by a respectable margin. This is in accordance by the fundamental reasoning behind the development of ensemble methods. Overall, we would pick the RFC over the other methods. Its superior performance (weighted mean of train and test) aligned with its efficiency both during training and testing phase are factors that were taken into consideration. The least efficient method in our study was SVM. This method was extremely slow during the training phase relative to all the other methods that we tested and yielded the lowest accuracy score. The overall results that we have found in this study are comparable to that of the publisher of the dataset [1]. Their method, however, uses a deep neural network and achieves an accuracy score of ~94%.

References

[1] Kachuee, Mohammad, Shayan Fazeli, and Majid Sarrafzadeh. "ECG Heartbeat Classification: A Deep Transferable Representation." *arXiv preprint arXiv:1805.00794* (2018).