Data Science

Automatic detection of Atrial Fibrillation (AF) episodes
Primary Topic: Data Mining
Course: 2021-1B – Group: 065 – Submission Date: 2022-02-06

Rushvanth Bhaskar
University of Twente
Netherlands
r.bhaskar@student.utwente.nl

Sainivedhitha Arunajatesan
University of Twente
Netherlands
s.arunajatesan@student.utwente.nl

ABSTRACT

The objective of this project was to detect the episodes of Atrial Fibrillation (AF) that last for a minimum period of 30 seconds automatically from the pre-processed ECG data. This data was provided by Erasmus Medical Centre in Rotterdam. This data consisted of certain periods classified as AF and non-AF by the healthcare professionals. Machine learning techniques such as linear SVM, non-linear SVM, Decision tree, Logistic Regression, K-NN, and Random Forest were run in Python to classify AF data. The performance was measured using accuracy, ROC−AUC score, precision, accuracy, recall and F1 score. Further, the ECG data provided has imbalanced data for the classes. To improve the quality of classification, two techniques were used to deal with class imbalance - SMOTE and weighted class models. Among the various classifiers that were used, Random Forest was found to have the best metrics. SMOTE was found to be the better technique to deal with class imbalance.

KEYWORDS

Atrial Fibrillation, Machine Learning, AF Detection, Class Imbalance, SMOTE, weighted class model, Data Mining, SVM, Decision Tree, K-NN, RF, Logistic Regression, ML Classifiers

1 INTRODUCTION

Atrial fibrillation (AF) is an irregular and often very rapid heart rhythm (arrhythmia) that can lead to blood clots in the heart. AF increases the risk of stroke, heart failure and other heart-related complications. AF is characterised by irregular ventricular rate (QRS complexes), absence of discrete P waves, replaced by irregular, chaotic F waves and long-short R-R cycles in the ECG signals. Manual electrocardiogram (ECG) interpretation for diagnosis is time demanding because it needs a high level of knowledge, and is subject to inter- and intra-observer variability. Deep learning techniques could be exploited in order for robust arrhythmia detection models to be designed. In this paper, we aim to solve the problem of automatic detection of atrial fibrillation through the use Data Mining models. For this, we validate various Machine Learning (ML) techniques in the given dataset and select the most promising classifier. In addition to this, we deal with the issue of imbalanced dataset provided to make it feasible for implementing ML techniques. A description of the dataset, as well as a full explanation of the technique used to produce predictions using machine learning algorithms, are provided in the report's subsequent parts.

This paper is organized as follows. Section 2 states the research questions, followed by some background on the topic in Section 3 which includes related work in this field and class imbalance techniques. Section 4 expands on the methodology that was employed to answer the research question followed by Section 5 which contains the results that were obtained. These results are discussed in Section 6 followed finally by the conclusion in Section 7.

All the artifacts that are used in this paper are open source and available at - https://github.com/rushvanth/AFDetection

1.1 Dataset Description

The preprocessed ECG dataset used in the atrial fibrillation detection project was acquired from Erasmus Medical Centre in Rotterdam. A semi-automatic program analyzed the ECGs for annotation of the R peaks. R peak detection was manually audited by a physician and atrial fibrillation or other arrhythmia were labeled. This dataset contains data of 150.000 periods of half a minute of different patients. Around 36.000 of these periods are labeled as AF (referred as 1) by healthcare professionals and the remaining are labeled as non-AF (referred as 0).

2 RESEARCH QUESTION

As mentioned in Section 1, manual detection of AF episodes using is time consuming and requires domain knowledge. Thus, the research questions that this paper aims to answer are: -

- With the continuous 24-hour ECG data, to what extent can one automatically detect episodes of AF occurring at a minimum of 30 seconds?
- How can one effectively deal with the class imbalance problem that is associated with the given dataset?

To answer this research question, the following methods will be used: -

- The given data will be split into training and test sets in order to validate ML classifiers
- Various ML classifiers will be selected based on related work, and used to classify the data. Their performance will then be compared.
- Two methods to deal with class imbalance will be employed - SMOTE and Weighted Class ML models, and their performance will also be compared.

3 BACKGROUND

This section describes the concepts related to this project in a succinct manner.

3.1 Related Work

The study of algorithms and statistical models that educate machines to handle large volumes of data more effectively is known as machine learning. It's a type of Artificial Intelligence that creates algorithms based on data patterns and previous data correlations. To solve complicated issues, interpret patterns, and extract important insights from data, we employ machine learning algorithms. The detection of atrial fibrillation in ECG has been discussed in various literature. Sraitih. et al. compared Support Vector Machine (SVM), Random Forest (RF) and K-Nearest Neighbour (K-NN) on ECG data and came to a conclusion that SVM outperformed the others with an accuracy of 83%. Further the computational costs were too less [13]. Lagerholm. et al. proposed Self Organizing Neural Networks to classify AF. Misclassification was too low (1.5%) in SOM compared to Conventional Cross Correlation method (4.4%). The CPU consumption was less than 1 min/record and the accuracy was 99.7% [5]. While comparing K-NN and RF, RF had a greater specificity and sensitivity of 98.3% and 92.8% respectively [4]. Nine different algorithms based on analysis of RR Intervals (RRI) and Atrial Activity were tested by Larburu et al. Out of these nine algorithms, the highest sensitivity of 97.64% was found in the statistical framework combination (using RRI). Kolmogorov Smirnov test (using RRI) showed highest specificity and lowest error of 96.08% and 5.32% respectively. Highest PPV was found in the combination of RRI and Atrial Activity (using RRI + PWA) to be 92.75%. In contrast, the algorithm using only Atrial Activity had the lowest sensitivity, lowest specificity and highest error [6]. The use of Long Short Term Memory (LSTM) on ECG data to identify AF resulted in a sensitivity of 97.87% and specificity of 99.29% [10]. A 16 layer 1D-CNN was proposed and tested against recurrent neural network and spectrogram learning. The accuracy of AF detection was 82%, 72% and 78% respectively [14]. An architecture was proposed with template based P wave absence detection, heart rate irregularity identification using Markov Process and Atrial Activity detection using Discrete Packet Wavelet Transform & QRS-T Cancellation. The specificity and sensitivity of this architecture was 96.09% and 93.80% respectively [2]. Nurmaini et al. fused Discrete Wavelet Transform (DWT) with 1D-CNN, incorporated various techniques to rectify class imbalance and obtained the following results: For detection of AF and normal sinus rhythm (NSR), only DWT had an accuracy of 92.97%, sensitivity of 87.46% and specificity of 87.46%. Using 10 fold for class imbalance along with DWT produced an accuracy 99.98%, sensitivity of 99.91% and specificity of 99.91%. For detection of AF, NSR and non-AF, DWT with 10 fold reflected an accuracy of 99.17%, sensitivity of 98.90% and specificity of 99.17% [7]. The use of SVM on Photoplethysmogram (PPG) data to detect AF resulted in a sensitivity of 94.2%, specificity of 96.2% and an accuracy of 95.7% [12]. The F1 score of K-margin-based Residual-Convolution-Recurrent neural network along with skewness-driven dynamic augmentation method for class imbalance was 0.8125 [15]. A novel algorithm was proposed by Peimankar et al. in which four classifiers such as RF, SVM, Adaptive Boosting (AdaBoost) and Group Method of data Handling (GMDH) were trained separately using 5-fold cross validation and the outputs were combined using Dempster-Shafer theory (DST). The accuracy and sensitivity of the proposed algorithm were 96.46% and 94% respectively for slightly longer episodes (60 beats per segment) of AF [9]. Rajesh et al. compared four different ML techniques (AdaBoost, SVM, Linear Discriminant Analysis, k-NN) to detect AF. Out of these four, AdaBoost had the highest specificity, sensitivity, Receiver Operating Characteristics (ROC) and accuracy [11].

3.2 Class Imbalance

An imbalanced classification problem is one in which the distribution of instances across recognized classes is uneven or biased. Unfortunately, when a classifier is trained on a dataset in which one of the response classes is rare, they can underestimate the probability of observing a rare event. To overcome this, there are several class imbalance techniques. One such technique is known as Synthetic Minority Over-sampling Technique (SMOTE) [1] which aims to solve this problem by a combination of under sampling the majority class and over sampling the minority class. Over sampling augments the original data by extrapolating new data in the minority class. Hatamian et al. compared three class imbalance techniques, oversampling, Gaussian Mixture Models (GMMs) and Deep Convolutional Generative Adversarial Networks (DCGAN) and concluded that DCGAN has a better AF classification accuracy in average than the other two methods [3]. A 11-layer deep CNN model has been proposed along with SMOTE to classify AF in ECG database. The accuracy of this model was found to be 98.3% [8]. Rajesh et al. suggested 3 steps for class imbalance: Re-sampling, SMOTE followed by distribution based data sampling. Solving class imbalance using these steps provided an accuracy of 98.3% [11]. Nurmaini et al.compared different class imbalance techniques. The paper used Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in 6 different combinations (DNN, CNN, RNN, RNN with Random Oversampling (ROS), RNN with SMOTE, CNN, CNN with 10 fold cross validation). Among these combinations, CNN with 10 fold showed the best results [7].

4 METHODOLOGY

The methodology employed in this paper involves using a number of classifiers on the dataset and comparing their performance. ML Classifiers are selected based on the usage in related work. In the end, 6 classifiers are used to classify the data - K-Nearest Neighbours, Decision Tree, Random Forest, Support Vector Machine (SVM) with Linear Kernel, SVM with RBF kernel, and Logistic Regression. First, the pre-processed data is taken as-is and the above mentioned classifiers are used to classify the data and their performance is recorded. Next, various class imbalance techniques are employed to enrich the dataset and the same ML classifiers are again trained and fitted on this new data. This performance is also recorded and the results are compared with the previous baseline results. Further sections will go into detail about these various steps.

4.1 Gathering Metadata

Before ML classifiers are used on the data, it is important to get an idea of the composition of the data itself and its properties. A visual inspection of the data shows that there are 30 features (columns) named [data1,...,data30] and the resulting class named Control. The Control column has values '0' or '1' which indicate Non-AF and AF

data respectively. The number of AF and Non-AF data samples are then counted. This shows that in total there are 113463 samples for Non-AF, and 36537 samples for AF. This is a ratio of 0.756 : 0.244.

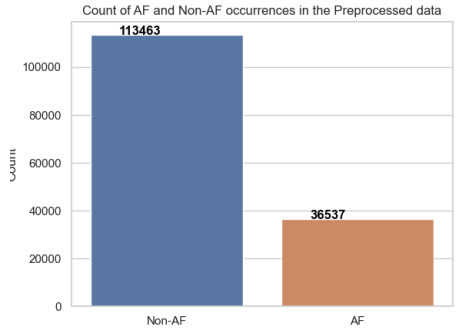

Figure 1: Count of AF and Non-AF Occurrences

An important phase during discovery of the dataset is the relationship between the features. This can also be termed as "correlation". The logic is that not all features are important for the final classification and that some features may be correlated to each other. This is helpful because it allows classification using fewer features. If a feature does not contribute to the final outcome, it can dropped. Similarly, if a feature is highly correlated with another, either of them can be chosen. The correlation matrix for the given dataset is visualized in Figure 2. From the correlation matrix, it is observed that the column `data4` and `data3` have a high correlation (0.51 and 0.47 respectively) with the outcome which mark them as important features.

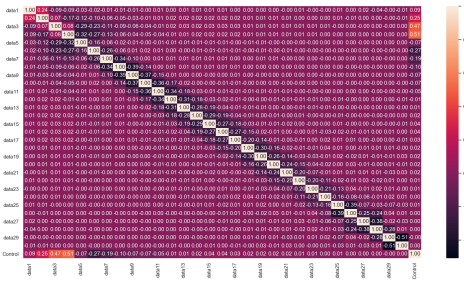

Figure 2: Correlation Matrix

Lastly, the data is split into training and test sets. This is done to make sure that the classifiers that are trained using the training data can then be used to classify the test set to measure their performance. 80% of the data is used to train each classifier and the rest 20% of the data is then classified and the results are validated.

4.2 Measuring Learning Times & Validating Classifiers

One of the most computationally intensive tasks in this research is the training of classifiers on the pre-processed AF data. Therefore, it is imperative to measure the learning times of various algorithms. In the end, this will also be a factor in determining the best classifier. In order to measure the learning times, various classifiers are trained on varying subsets of the data, and their performance is measured. This measures how well a certain classifier performs on various amounts of data. If a classifier achieves a high accuracy with a small amount of data, it should be preferred. The scalability of the model is also measured which is the fit times on varying amount of samples. Learning curves are only plotted for K-NN, SVM (Linear kernel), SVM (RBF kernel), and Random Forest models. This is because these models gives us a good coverage of different types of classifiers - linear, non-linear, neighbour clustering based, and tree based. The learning curves are plotted for 4 classifiers and are visualized in Figure 3.

Additionally, validation curves are also plotted for these classifiers. The validation curves are helpful to visualize whether a model is underfitting or overfitting. Overfitting is a phenomena where the classifier is trained too well. This can be detrimental in some cases where there is noise in the dataset that is picked up as a feature by the classifier. Underfitting is the opposite where a classifier is unable to model the data and hence fails to predict new data. Both of these are undesirable and should they , corrective measures should be taken. The validation curves for the classifiers are visualized in Appendix B.

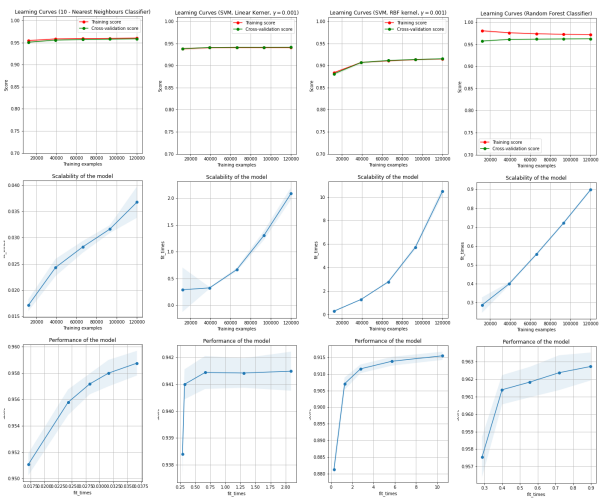

Figure 3: Learning Curves

4.3 Classifying Data using ML Models

Now that there is a good grasp of what the makeup of the data, we can establish a baseline for performance of the various classifiers by fitting and scoring them on the training data. For this purpose, 6 different classifiers are trained and fitted on the data. During this process, various metrics are collected like the training times and fit times. These can later be used to compare the performance of the models. For the Logistic Regression model, the number of maximum iterations are changed from the default value of 100 to 1000. This

is because the `lbgfs` solver could not converging at 100 iterations. Similarly, a maximum depth of 5 was specified for the Decision Tree Classifier. This is to prevent overfitting of the model by performing more iterations than necessary. All other classifiers were used with their default parameters. As an extension to splitting the data into training and testing sets, cross validation was performed with 5 folds. A K-Folds cross-validator provides train/test indices to split data into train/test sets. The data is then split into k consecutive folds and each fold is then used once as a validation while the k - 1 remaining folds form the training set. This is to ensure that the models are not trained with the same training and test data each time. For measuring the performance of the classifiers, five metrics are used - Accuracy, Precision (Weighted), Recall (Weighted), F1 Score (Weighted), and ROC-AUC score.

Accuracy is the most common metric for a classifier. This is the fraction of correct samples predicted by the classifier. Recall, also known as sensitivity, is the fraction of positive events that were predicted correctly. In conjunction to recall, precision defines the fraction of positive events that were actually positive. The F1 score then takes the last 2 metrics and is defined as the harmonic mean of the recall and precision. The ROC-AUC score is an extremely helpful metric in understanding the balance between true-positive rate and false positive rates.

4.4 Dealing with Class Imbalance

As demonstrated in Section 4.1, there is an inherent class imbalance in the pre-processed data. There are much more samples for Non-AF than AF. The disadvantages of such a dataset is already discussed in Section 3.2. To deal with this class imbalance, 2 techniques are compared. First is the use of class weights in classifiers. Adding weights to classes when training and fitting data instructs the classifier to give much more importance to one class over the other. This makes it such that misclassifying the heavier class will result in a larger penalty. The issue here is determining the class weights for each class. While it is intuitive that the minority class should receive a larger weight, finding the optimum weights for both classes requires a grid search. A grid search basically takes a wide range of class weights, trains the model for each pairs of weights and records the performance. In the end, the pair of weights that resulted in the best performance are chosen as class weights. A Logistic Regression classifier is chosen for the grid search and the weights are all values in the Numpy linear space from 0.0 - 0.99 with 200 as interval. This gives us 200 weights for the Logistic Regression classifier. This particular classifier is chosen because it is computationally less expensive. Once the optimum weights are identified, they can be applied to the other classifiers too. The results of the grid search will be discussed in Section 5.

Apart from weighted models, SMOTE was also used to address the class imbalance problem. SMOTE works by generating new samples of the minority class. This is done by creating synthetic data that is mathematically close to data in the minority class. More specifically, it selects samples that are close in feature space and drawing a line between these samples. This line is then extrapolated and new samples are points that fall on this extended line. Along with oversampling the minority data, random under sampling of the majority data is also performed. This makes it so that there is a balance between the samples of each class. After performing SMOTE on the data, the classifiers are used again on this enriched data and the performance is compared to before.

5 RESULTS

In this section, the results of the various classifiers are discussed.

5.1 ML Models - Baseline Performance

When the classifiers were trained on the pre-processed data as described in Section 4.3, the Random Forest model scored the highest accuracy with 96.33 followed by SVM (RBF kernel) which scored an accuracy of 96.20. All other models were close behind with K-NN scoring 95.92, Linear SVM scoring 94.08, Decision Tree scoring 93.86, and Logistic Regression scoring 93.30. Figure 5 visualizes the various metrics that were measured. The importance of features are also visualized for Decision Trees and Linear SVM. Non Linear SVMs transform the data into higher dimension spaces and thus it is not possible to compute feature importance. It is interesting to note that Decision tree considers `data4` as them ost important feature while Linear SVM considers `data1` to be the most important feature.

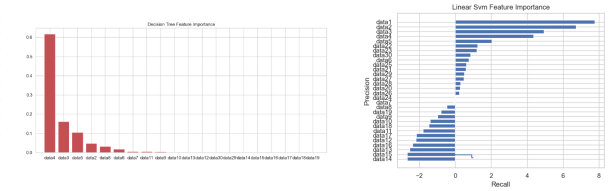

Figure 4: Feature Importance

5.2 Weighted ML Models

To determine the optimum weights for the majority and minority class, a grid search was performed using Logistic Regression classifier. 200 weights at equal intervals between 0.0 to 0.99 were taken and assigned to one class. The other class was assigned the complementary value of 1-(weight). Each time, the classifier was trained and fitted and the ROC-AUC score was considered for scoring. Therefore, the weights that produced the best ROC-AUC score were considered as optimum. The reason for selecting ROC-AUC score over accuracy is discussed in Section 6. The results of grid search produced optimum weights of 0.06964824120603015 for the Non-AF (0) class and 0.9303517587939698 for the AF (1) class. This produced the highest ROC-AUC score of 96.533. The different weights and their corresponding scores are visualized in Figure 6.

Grid search for all classifiers was not performed due to extremely high computation demand of the grid search operation. After the optimum weights were discovered, all the classifiers except K-NN were re-run on the pre-processed data. K-NN does not support providing class weights and hence it was skipped. The results are visualized in Figure 5. It is observed that the accuracy score of different classifiers shows a sharp decrease compared to before. Possible reasons for this are discussed in the upcoming section.

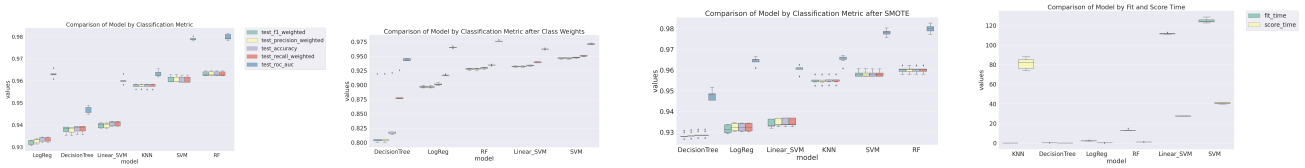

Figure 5: Classifier Results & Fit Times

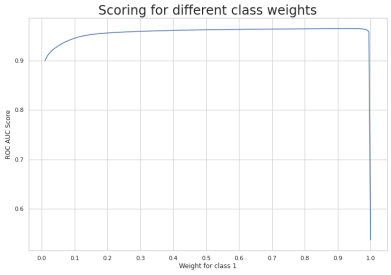

Figure 6: Weights Grid Search

Apart from weighted classes, SMOTE was also employed to enrich the data as described in Section 3.2. After performing SMOTE, the distribution of data in the classes were more balanced and is shown in Figure 7.

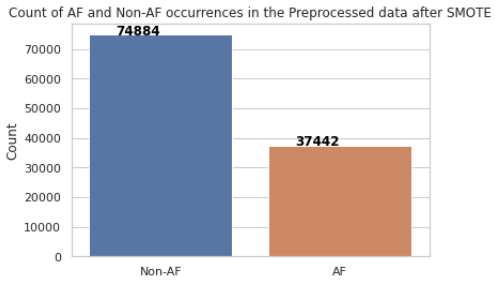

Figure 7: Count of AF and Non-AF Occurrences after SMOTE

The results of running the classifiers on the data after performing SMOTE is shown in Figure 5. In general, it is seen that accuracy is similar to before, but the other metrics improve. Detailed metrics for all the various runs are available in the Appendix.

6 DISCUSSION

In this section, the results obtained in Section 5 are discussed and possible reasons are given as to why certain results were observed.

In general, it is seen that all the classifiers are able to classify data with a high accuracy. This is mostly in part because of the quality of the given data. The pre-processed data is extremely well defined with no undefined values. This allows the classifiers to get a high accuracy score. But it is important to note that in imbalanced datasets such as these, only looking at accuracy scores can be deceiving. The classifier can get a high accuracy score by simply classifying all samples as the majority class. Therefore, it is important to look at the other metrics like recall and ROC-AUC scores. As expected, it is apparent from the results in Figure 5 that these are lower for the AF class. The best performing classifier was found to be Random Forest followed by SVM with an RBF kernel. But from the graph that depicts fit times in Figure 5, it is seen that SVMs take much longer than Random Forest to train and fit the data. This makes Random Forest classifier much more lucrative. It can also be seen that the K-NN model also takes a long time to score the data.

SMOTE is preferred to weighted models when dealing with class imbalances because SMOTE is classifier agnostic in the sense that it can be used alongside any classifier. Finding the optimum class weights for a classifier can be extremely computationally intensive and as is seen from the results, these weights are not same for all classifiers. The optimum pair of weights for one classifier need not be the best weights for another classifier. On the other hand, SMOTE provided the increase in metrics like precision, recall, and ROC-AUC score that was expected. Even though accuracy may have dropped a bit, in the case of this problem, it is much more safer to misclassify a sample to class Non-AF than class AF. If a sample that is AF is classified as Non-AF, a physician can review the patient and determine that it is a false positive. But if an AF sample is misclassified as Non-AF, it could cause fatalities. Thus, the penalty for a false negative must be much higher.

7 CONCLUSION

In conclusion, to answer the research questions from Section 2, it is possible to detect AF episodes with a high degree of accuracy given data that is pre-processed to remove outliers and anomalies. The pre-processed data that was provided allowed easy application of Machine Learning Classifiers and the metrics show that they can be used with a high degree of confidence. SMOTE was found to be the better technique to deal with class imbalance as it was classifier agnostic and can be applied to any dataset. It is also less computationally demanding than weighted class models. The Random Forest classifier was found to have the highest accuracy and is preferred to SVMs which have high metrics but take longer to train and score. Since Random Forest classifiers are a group of decision trees, tree based classifiers are concluded to have the best performance on this dataset. For future work, optimum weights for every classifier can be calculated and Neural Networks can be used to classify the data.

REFERENCES

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (Jun 2002), 321–357. https://doi.org/10.1613/jair.953

[2] R. Couceiro, P. Carvalho, J. Henriques, M. Antunes, M. Harris, and J. Habetha. 2008. Detection of Atrial Fibrillation using model-based ECG analysis. In *2008 19th International Conference on Pattern Recognition*. 1–5. https://doi.org/10.1109/ICPR.2008.4761755

[3] Faezeh Nejati Hatamian, Nishant Ravikumar, Sulaiman Vesal, Felix P. Kemeth, Matthias Struck, and Andreas Maier. 2020. The Effect of Data Augmentation on Classification of Atrial Fibrillation in Short Single-Lead ECG Signals Using Deep Neural Networks. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1264–1268. https://doi.org/10.1109/ICASSP40776.2020.9053800

[4] Alan Kennedy, Dewar D. Finlay, Daniel Guldenring, Raymond R. Bond, Kieran Moran, and James McLaughlin. 2016. Automated detection of atrial fibrillation using R-R intervals and multivariate-based classification. *Journal of Electrocardiology* 49, 6 (Nov. 2016), 871–876. https://doi.org/10.1016/j.jelectrocard.2016.07.033

[5] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo. 2000. Clustering ECG complexes using Hermite functions and self-organizing maps. *IEEE Transactions on Biomedical Engineering* 47, 7 (2000), 838–848. https://doi.org/10.1109/10.846677

[6] N Larburu, T Lopetegi, and I Romero. 2011. Comparative study of algorithms for Atrial Fibrillation detection. In *2011 Computing in Cardiology*. 265–268.

[7] Siti Nurmaini, Alexander Edo Tondas, Annisa Darmawahyuni, Muhammad Naufal Rachmatullah, Radiyati Umi Partan, Firdaus Firdaus, Bambang Tutuko, Ferlita Pratiwi, Andre Herviant Juliano, and Rahmi Khoirani. 2020. Robust detection of atrial fibrillation from short-term electrocardiogram using convolutional neural networks. *Future Generation Computer Systems* 113 (2020), 304–317. https://doi.org/10.1016/j.future.2020.07.021

[8] Saroj Kumar Pandey and Rekh Ram Janghel. 2019. Automatic detection of arrhythmia from imbalanced ECG database using CNN model with SMOTE. *Australasian Physical & Engineering Sciences in Medicine* 42, 4 (Nov. 2019), 1129–1139. https://doi.org/10.1007/s13246-019-00815-9

[9] Abdolrahman Peimankar and Sadasivan Puthusserypady. 2018. Ensemble Learning for Detection of Short Episodes of Atrial Fibrillation. In *2018 26th European Signal Processing Conference (EUSIPCO)*. 66–70. https://doi.org/10.23919/EUSIPCO.2018.8553253

[10] Georgios Petmezas, Kostas Haris, Leandros Stefanopoulos, Vassilis Kilintzis, Andreas Tzavelis, John A Rogers, Aggelos K Katsaggelos, and Nicos Maglaveras. 2021. Automated Atrial Fibrillation Detection using a Hybrid CNN-LSTM Network on Imbalanced ECG Datasets. *Biomedical Signal Processing and Control* 63 (2021), 102194. https://doi.org/10.1016/j.bspc.2020.102194

[11] Kandala N.V.P.S. Rajesh and Ravindra Dhuli. 2018. Classification of imbalanced ECG beats using re-sampling techniques and AdaBoost ensemble classifier. *Biomedical Signal Processing and Control* 41 (2018), 242–254. https://doi.org/10.1016/j.bspc.2017.12.004

[12] Shih-Ming Shan, Sung-Chun Tang, Pei-Wen Huang, Yu-Min Lin, Wei-Han Huang, Dar-Ming Lai, and An-Yeu Andy Wu. 2016. Reliable PPG-based algorithm in atrial fibrillation detection. In *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 340–343. https://doi.org/10.1109/BioCAS.2016.7833801

[13] Mohamed Sraitih, Younes Jabrane, and Amir Hajjam El Hassani. 2021. An Automated System for ECG Arrhythmia Detection Using Machine Learning Techniques. *Journal of Clinical Medicine* 10, 22 (Nov. 2021), 5450. https://doi.org/10.3390/jcm10225450

[14] Zhaohan Xiong, Martin Stiles, and Jichao Zhao. 2017. Robust ECG Signal Classification for the Detection of Atrial Fibrillation Using Novel Neural Networks. In *Computing in Cardiology Conference (CinC)*. Computing in Cardiology. https://doi.org/10.22489/cinc.2017.066-138

[15] Yuxi Zhou, Shenda Hong, Junyuan Shang, Meng Wu, Qingyun Wang, Hongyan Li, and Junqing Xie. 2019. K-margin-based Residual-Convolution-Recurrent Neural Network for Atrial Fibrillation Detection. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2019/839

APPENDIX A - DENSITY CURVES

When determining feature importance, it is beneficial to plot density curves for the features. The density curves show the importance of each feature to the outcome. In Figure 8, features which have separation between the red and green lines imply that the feature is important to the outcome.

Figure 8: Density Curves

APPENDIX B - VALIDATION CURVES

As described in Section 4.2, validation curves show whether the classifier is overfitting or underfitting.

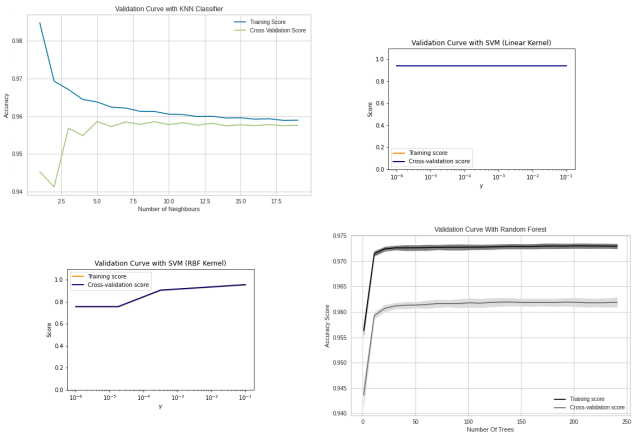

Figure 9: Validation Curves

APPENDIX C - ML CLASSIFIER METRICS

Apart from accuracy and other metrics, there are some interesting metrics that are of relevance. These metrics are included for every classifier. Note - These metrics are for classifiers that were run on the pre-processed data without any class imbalance techniques.

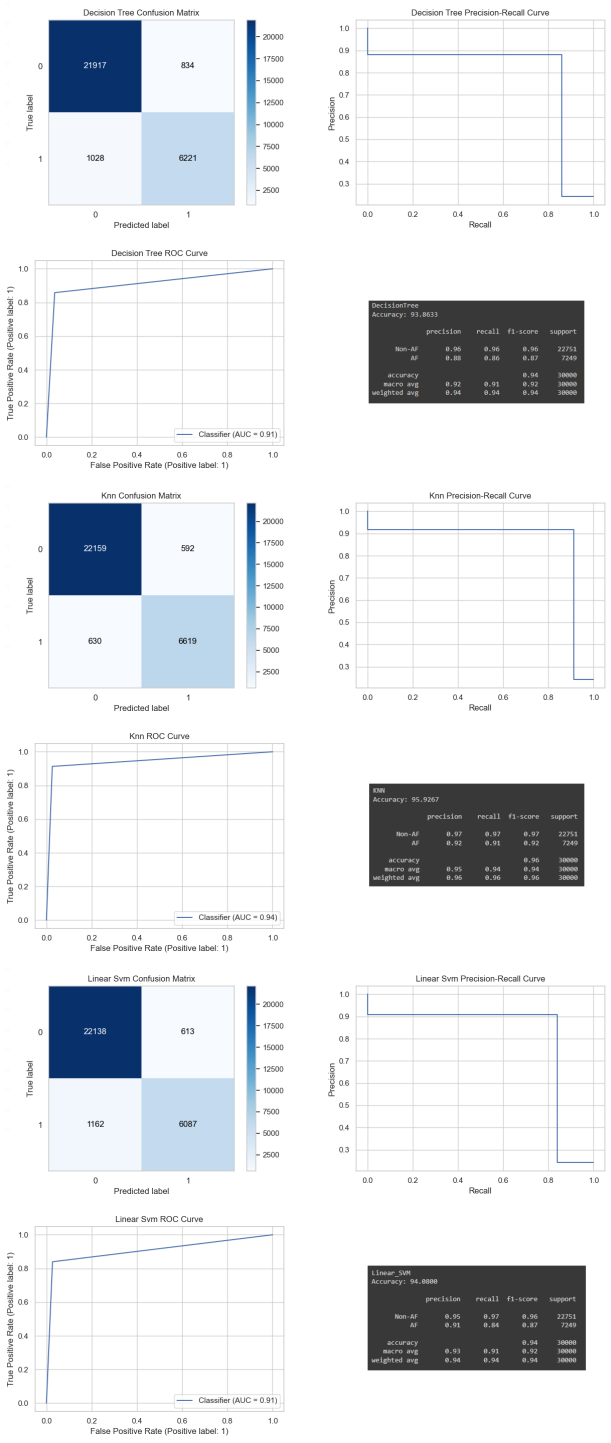

Figure 10: Classifier Metrics without Class Imbalance Techniques

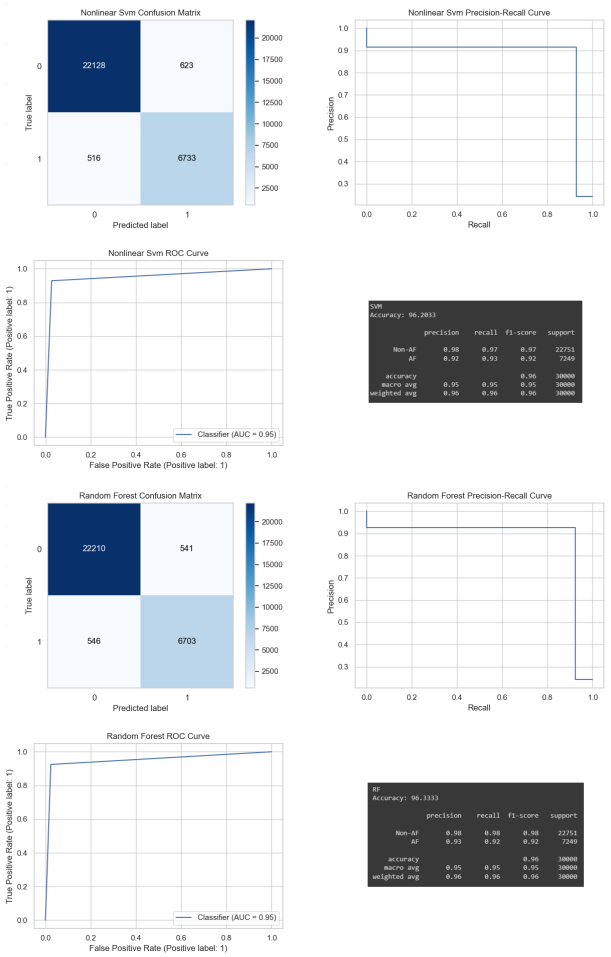

Figure 11: Classifier Metrics without Class Imbalance Techniques

APPENDIX D - ML CLASSIFIER METRICS - SMOTE

SMOTE was used to oversample the minority class as mentioned in Section 3.2. The results after using SMOTE are shown in Figure 12.

APPENDIX E - ML CLASSIFIER METRICS - WEIGHTED CLASS MODELS

After using class imbalance techniques, the classifiers were run again. The results are depicted in Figure 13.

DecisionTree
Accuracy: 80.2433

	precision	recall	f1-score	support
Non-AF	0.98	0.75	0.85	22751
AF	0.55	0.96	0.70	7249
accuracy			0.80	30000
macro avg	0.77	0.86	0.78	30000
weighted avg	0.88	0.80	0.82	30000

LogReg
Accuracy: 89.7833

	precision	recall	f1-score	support
Non-AF	0.98	0.88	0.93	22751
AF	0.72	0.95	0.82	7249
accuracy			0.90	30000
macro avg	0.85	0.92	0.87	30000
weighted avg	0.92	0.90	0.90	30000

RF
Accuracy: 92.9733

	precision	recall	f1-score	support
Non-AF	0.98	0.93	0.95	22751
AF	0.80	0.94	0.87	7249
accuracy			0.93	30000
macro avg	0.89	0.93	0.91	30000
weighted avg	0.94	0.93	0.93	30000

SVM
Accuracy: 94.9600

	precision	recall	f1-score	support
Non-AF	0.98	0.95	0.97	22751
AF	0.86	0.95	0.90	7249
accuracy			0.95	30000
macro avg	0.92	0.95	0.93	30000
weighted avg	0.95	0.95	0.95	30000

Linear_SVM
Accuracy: 93.2567

	precision	recall	f1-score	support
Non-AF	0.98	0.93	0.95	22751
AF	0.81	0.95	0.87	7249
accuracy			0.93	30000
macro avg	0.89	0.94	0.91	30000
weighted avg	0.94	0.93	0.93	30000

Figure 13: Classifier Metrics with Weighted Class

DecisionTree
Accuracy: 92.8514

	precision	recall	f1-score	support
Non-AF	0.93	0.96	0.95	15026
AF	0.91	0.86	0.89	7440
accuracy			0.93	22466
macro avg	0.92	0.91	0.92	22466
weighted avg	0.93	0.93	0.93	22466

LogReg
Accuracy: 93.3099

	precision	recall	f1-score	support
Non-AF	0.93	0.97	0.95	15026
AF	0.94	0.85	0.89	7440
accuracy			0.93	22466
macro avg	0.93	0.91	0.92	22466
weighted avg	0.93	0.93	0.93	22466

RF
Accuracy: 96.0518

	precision	recall	f1-score	support
Non-AF	0.97	0.97	0.97	15026
AF	0.94	0.94	0.94	7440
accuracy			0.96	22466
macro avg	0.96	0.95	0.96	22466
weighted avg	0.96	0.96	0.96	22466

KNN
Accuracy: 95.5444

	precision	recall	f1-score	support
Non-AF	0.96	0.97	0.97	15026
AF	0.94	0.92	0.93	7440
accuracy			0.96	22466
macro avg	0.95	0.95	0.95	22466
weighted avg	0.96	0.96	0.96	22466

SVM
Accuracy: 95.7180

	precision	recall	f1-score	support
Non-AF	0.97	0.97	0.97	15026
AF	0.94	0.93	0.94	7440
accuracy			0.96	22466
macro avg	0.95	0.95	0.95	22466
weighted avg	0.96	0.96	0.96	22466

Linear_SVM
Accuracy: 93.3989

	precision	recall	f1-score	support
Non-AF	0.93	0.97	0.95	15026
AF	0.93	0.86	0.90	7440
accuracy			0.93	22466
macro avg	0.93	0.92	0.92	22466
weighted avg	0.93	0.93	0.93	22466

Figure 12: Classifier Metrics with SMOTE