Utilising machine learning to investigate electrical activity in ventricular myocytes

J.Francis

with thanks to

Dr. J.Tabak , Dr. Y.Hill , Prof. C.Soeller , G.L.Murphy

A dissertation submitted in partial fulfilment of the requirements for the degree of

Applied Data Science and Statistics MSc

University of Exeter

11 September 2020

I confirm that the work submitted is my own and that appropriate credit has been given where reference has been made to the work of others.

Table of Contents

ABSTRACT

Early afterdepolarizations (EADs) are denoted by abnormal repolarization in phase 2 or 3 of the cardiac action potential. Propagation of EADs in ventricular myocytes are one of the primary mechanisms causing lethal ventricular arrhythmias. Therefore, it is essential that we understand the mechanisms that give rise to EAD genesis. A novel machine-learning based approach is proposed to investigate the effect that changing multiple parameters has on the transition from normal action potentials to EAD dynamics. Firstly, a database was created by G.L. Murphy which exhaustively mapped the conditions under which normal action potentials and EADs arise during bradycardia. From this database, a dataset of 212, 249 simulations with 8 varied parameters as predictor variables, were used to predict EAD class (EAD vs non-EAD). Multiple models were trained with different data sampling techniques (to account for class imbalance) to discover the best performing model for EAD classification. Finally, SHAP values were used to understand the importance of parameters' roles in EAD genesis. XGBoost (extreme gradient boosting) proved to be an excellent classifier, outperforming random forests, decision trees and logistic regression, misclassifying only 0.07% of unseen data. SHAP values revealed that conductance of rapid activating delayed rectifier currents and permeability of L-type calcium currents were the top 2 parameters in predicting EAD genesis - confirming previous research. Beyond the scope of this project, further exploration with SHAP values should reveal interactions between parameters. Future research can also use this machine learning approach to explore other biophysical dynamics that lead to arrhythmias.

INTRODUCTION

Role of EADs in arrhythmias

Through extensive research, it has been determined that early afterdepolarizations (EADs), as well as antiarrhythmic treatments can lead to arrhythmias [1][2]. These arrhythmias, otherwise known as dysrhythmias, refer to when the heart pulsates irregularly, whether it's the rate of the heartbeat or the associated rhythm. Early afterdepolarizations (EADs) are abnormal oscillations during phase 2 or phase 3 of an action potential [3]. These EADs can prompt the formation of new APs that can be perceived as "ectopic beats" when viewed on an electrocardiogram [4].

Almost five hundred 500 years ago, undamaged cardiac tissue containing EADs were first designated as a form of 'triggered activity' [3]. Afterwards, EADs were incriminated as the primary means by which Long QT Syndromes (LQTS), whether acquired at some point throughout one's lifetime or inborn, were promoted [3]. EAD propagation is one of the main mechanisms causing long QT syndromes (LQTS). LQTs prolongs the action potential (during the repolarization phase) and as a result can cause arrhythmias such as Torsades des Pointes (TdP). In these earlier days, electrophysiological studies were also carried out which involved a mixture of patients previously diagnosed with either innate LQTSs or LQTSs contracted later on in life. Based on endocardial electrogram recordings, it was found that the T-U waves and generators of arrhythmias are actually due in part to EADs [4]. There isn't a significant amount of study regarding this topic; however, from those studies that were carried out, researchers are now confident that the cause of arrhythmias at the myocardial tissue level are in fact caused by EADs at the myocyte level [3].

The most successful methods to treat arrhythmias are rather invasive surgical treatments and the majority of those patients who will have fatal arrhythmias will not show the warning signs needed for surgical treatments such as catheter ablation and implantable cardioverter defibrillators. Hence it is important to find alternative treatments such as less invasive drug therapies. However, these are not advanced enough to always be the preferred method of treatment. For instance, blocking one channel through drug treatment may cause adverse side effects, potentially causing another arrhythmia or another complication [2].

In the course of the 1970s and 1980s, the thought that noncardiovascular drugs potentially extend the QT interval and prompt Torsades de Pointes (TDP) or unexpected death was proposed [2]. According to Kannankeril et al. [2], congenital Long QT Syndrome could develop from alterations disrupting ion currents and the method through which drugs give rise to LQTS acquired after birth is, majority of the time, a block of I_{Kr}. I_{Kr} can be blocked by drugs such as antiarrhythmics, antipsychotics, antibiotics and antihistamines [2].

From information presented above, we can safely come to the conclusion that there exists a substantial amount of evidence supporting the idea that both EADs and antiarrhythmics can promote arrhythmias through a variety of ways and it's therefore important to study the ionic mechanisms that lead to EADs so that we can find ways to abolish them.

Ventricular action potential

The activation and deactivation of ion currents play a major role in the cardiac action potential. Influxes and effluxes of ions via ion currents (I_{ion}) depolarize and repolarize the cell membrane, dictating the shape and nature of the action potential. Therefore, it is important to understand the conductance and movement of the inward and outward currents that are responsible for a normal ventricular action potential before studying an abnormal action potential such as an EAD. A ventricular action potential can be characterized by five distinct phases and the major currents that dictate its shape:

Phase 0 (Depolarization) – An actional potential from a neighbouring cell such as a pacemaker cell or another ventricular myocyte can cause the membrane potential in a single ventricular myocyte to reach a threshold value of ~ -70 mV. Once this threshold value is reached, the cell membrane becomes less permeable to potassium ions (K^+) and the efflux of K^+ slows down. An increase in inward L-type calcium currents ($I_{Ca,L}$) through long-lasting calcium channels allow for an influx of calcium ions (Ca^{2+}). As sodium conductance (G_{Na}) increases, fast sodium channels also open and there is a rapid influx of sodium ions (Na^+) into the cell via inward sodium currents (I_{Na}). The cell is rapidly depolarized; the membrane potential shifts away from the potassium equilibrium (E_K) of ~ -90 mV and towards the sodium equilibrium (E_{Na}) of ~ +50 mV.

Phase 1 (Initial repolarization) – Decreased G_{Na} cause sodium channels to rapidly close, decreasing I_{Na} and hence, the movement of Na^+ into the cell. A brief increase in potassium conductance (G_K) make specific transient potassium channels open. Transient outward potassium currents (I_{to}) rapidly activate and deactivate and result in a brief efflux of K^+. Therefore, the membrane potential becomes more negative.

Phase 2 (Plateau) – There is an influx of Ca^{2+} via $I_{Ca,L}$ and an efflux of K^+ via delayed rectifier potassium currents– slow activating delayed rectifier currents (I_{Ks}) and rapid activating delayed rectifier currents (I_{kr}).Calcium permeability (P_{Ca}) significantly increases and the Ca^{2+} influx and K^+ efflux balance each other in such a way that the membrane potential becomes relatively stable, slowly repolarizing and essentially plateauing.

Phase 3 (Final Repolarization) – As L-type calcium channels close, potassium channels remain open and potassium conductance predominates over the conductance of all other ions. The cell rapidly repolarizes and the membrane potential shifts toward the E_K largely due to the efflux of K^+ via the delayed rectifier potassium currents and inward rectifier currents (I_{K1}).

Phase 4 (Resting potential) –During this resting state, the delayed rectifier channels are closed, however, I_{K1} continue to flow through potassium channels, causing K^+ to leak out of the cell. The cell membrane is most permeable to K^+ and the Na^+/K^+ ion pump moves three sodium ions out of the cell and two potassium ions into the cell. As a result, the resting membrane potential typically remains between -80 to -90 mV.

Early afterdepolarizations (EADs)

Experiments and simulations of mathematical models have led to a great understanding of the EAD mechanisms. EADs can be triggered by an imbalance of inward and outward currents to the extent that the current required to complete the final repolarization phase (phase 3) is not met. This is called the reduced repolarization reserve [5]. It has been widely studied and proven that EAD genesis requires a reduction in repolarizing currents such as I_K and an increase in depolarizing currents such as Ca^{2+} transients [3]. Three currents have been widely cited as playing major roles in EAD genesis: outward delayed rectifier potassium currents (I_{Kr} and I_{Ks}), inward L-type calcium currents ($I_{Ca,L}$) and the inward sodium-calcium exchanger currents (I_{NCX}).

It should be stressed that in order for the current imbalance to reverse repolarization, triggering an EAD, it almost always needs to be in combination with the plateau phase (phase 2 of the action potential) moving beyond 0mV (in the negative direction) for a relatively significant amount of time. Otherwise, it is quite rare for the current imbalance to cause an EAD if the plateau phase remains above 0mV [2].

Methods such as multivariable regression [6] and bifurcation analyses [5][7] have led to crucial findings regarding EAD genesis. Inspired by the work of Fletcher et al. [8] and Ferrat et al. [9] we propose a new approach to understanding the ionic mechanisms that play key roles in EAD genesis.

Aims

This study aims to understand the effect that varying multiple conductances/permeability of ion currents has on EAD genesis using a novel approach. More specifically, using a database generated by G.L.Murphy, the aim is to firstly train a classification model to classify EADs and non-EADs with 8 conductance/permeability parameters as independent variables. Secondly, using the classification model and with the aid of SHAP values, we aim to understand the parameters' importance to EAD genesis.

** Please note that the independent variables used here are often referred to as 'parameters' in the biomedical domain and therefore, throughout the rest of this report, the terms parameters, independent variables/features will be used interchangeably.*

THE DATA

Computational modelling was chosen over experimental studies to build the database needed for this project. Biological systems are extremely complex and mathematical models have widely been used to simulate action potential dynamics in cardiomyocytes. Computational modelling is less time consuming than experimental studies, it enables simulations of action potential dynamics and allows us to alter parameters of a mathematical model quickly. Simulating these models can result in highly multidimensional parameter spaces [10] that we can exploit with machine learning algorithms.

The O'Hara Rudy Model (ORD)

The endothelial O'Hara Rudy model (ORd) [11] was chosen and simulated to create the dataset used in this study. It is a model of an undiseased human ventricular myocyte developed using experimental, undiseased, human ventricular data [11]. Using this data, O'Hara et al. [11] were able to produce ventricular action potential dynamics including EADs. The model was developed to understand the single cell electrical behaviour (including major currents and APs) underlying arrhythmias. Although a single EAD cannot cause an arrhythmia on its own, it is important to understand dynamics at the relatively basic cellular level before studying them at the tissue level.

This model was chosen because it is a recent human model, it takes into account 41 parameters [11] and most importantly, it's capable of producing EADs.

Creating the database

The database and hence the dataset used in this project was created by MSc student G.L.Murphy. Based on the work of Fletcher et al [8], he used a GPU-based modelling technique to generate the database. Murphy used MATLAB to generate the database of simulations with features of the electrical activity of a myocyte. Utilising GPUs allowed these simulations to be computed at rapid speed.

The ORd [11] model was verified by reproducing figures that matched those found in the ORd paper's supplementary text [11]. This model was then converted to an .ode file, a format compatible with XPPAUT [12]. The model in this .ode file was provided to clODE [13] and xppToolbox [14] in MATLAB. These tools convert the .ode file into a .CL file which represents all the ODE (set of ordinary differential equations), conversions and calculations of the model. Corresponding sets of initial conditions and parameter values were defined in accordance with the ones used in the ORd study [11] and this updated model was also validated by producing figures that matched those found in the ORd paper's supplementary text [11].

Eight parameters were varied and scaled independently using a multiplier ranging from 0.1-1.9. Only 8 parameters were varied due to time constraints and they were selected based on their identified importance to the EAD mechanisms discussed in the previous section or in an effort to discover new, unidentified EAD mechanisms. Cycle length of 2000ms was fixed to model bradycardia which has higher rates of EAD genesis [2]. Parameter sets were determined through uniformed random sampling to create each simulation. Simulations ran for 101 beats to reach steady state conditions but only the last 2 beats (4000ms) were recorded and analysed.

Filtering was then done to remove as many abnormalities as possible, excluding 1% of the data. Simulations with $V_{min} \geq -75\ mV$ and/or $V_{max} \leq -75\ mV$ indicated failed repolarisation and depolarisation failure respectively and were therefore removed using clODE *thres2 observer* [13]. Using *thres2 observer*, APD90 was also approximated as the time between an up threshold of 0.105 cell voltage and a down threshold of 0.095 cell voltage. Simulations with more than 2 up-thresholds indicated non-physiological behaviour and were hence removed. Peaks were defined as such if they had a minimum amplitude of $5\ mV$ and minimum separation of $70\ ms$. Simulations with more than 6 peaks indicated extreme and aperiodic behaviour and these simulations were also found and removed using the *findpeaks* function in MATLAB. Some simulations with abnormal behaviours made it through filtering and due to time constraints, all could not be identified and removed, however they only represent an extremely small portion of the data and were considered noise.

Model parameter	**Parameter control**	**Current abbreviation**	**Current**
G_{Na}	75 mS/μF	I_{Na}	Na^+ current
G_{to}	0.02 mS/μF	I_{to}	Transient outward K^+ current
P_{Ca}	0.0001 cm/s	$I_{Ca,L} + I_{Ca,Na} + I_{Ca,K}$	Ca^{2+}, Na^+ and K^+ currents ($I_{Ca,L}$, $I_{Ca,Na}$, $I_{Ca,K}$) through the L-type Ca^{2+} channel
G_{kr}	0.046 mS/μF	I_{kr}	Rapid delayed rectifier K^+ current
G_{Ks}	0.0034 mS/μF	I_{Ks}	Slow delayed rectifier K^+ current
G_{K1}	0.1908 mS/μF	I_{K1}	Inward rectifier K^+ current
G_{NCX} also referred to as G_{NaCa}	0.0008 mS/μF	$I_{NaCa} = I_{NaCa,i}+I_{NaCa,ss}$	Sodium calcium exchange current which is a sum of the myoplasmic and

			subspace components ($I_{NaCa,i}$ and $I_{NaCa,ss}$ respectively) of the Na^+/Ca^{2+} exchange current
G_{pCa}	0.0005 mS/μF	I_{pCa}	Sarcolemmal Ca^{2+} pump current

Table 1: Altered Conductance/Permeability Parameters used as the predictor variables
These parameters were varied and their corresponding currents and control values are shown here as well.

The dataset

The dataset used in this study, contains 212, 249 observations (simulations) and 11 variables. These variables are cycle length, APD90, class (the dependent variable) and 8 independent variables - G_{Na}, G_{to}, P_{Ca}, G_{Kr}, G_{Ks}, G_{K1}, G_{NCX}, G_{pCa}.

Cycle length and APD90 were removed before model fitting. Cycle length was a fixed variable and APD90 is a characteristic of the action potential, they were not varied model features. We are interested in classifying action potentials with the varied model conductances/permeability as predictor variables.

The independent variables are altered channel conductances/permeability used to predict the class variable and their values represent the scale factor at which they were altered. A scale factor of 1 represents the control value (Table 1). Scale factors below 1 represent a current block (decrease in conductance/permeability) and scale factors above 1 indicate an increase in currents (increased conductance/permeability). Table 1 provides more information on these independent variables.

Class was originally a numerical variable representing the average number of excess peaks present in a 2-beat window. These values were either 0, 0.5, 1, 1.5 or 2. Observations without EADs had a value of 0. Those with values of 0.5 or 1.5 were considered alternans. Observations with values of 1 or 2 had either 1 excess peak in each beat or 2 excess peaks in each beat respectively. Alternans represent 0.42% of the dataset and 37.74% of all the EADs in the dataset (Table 2).

Class was transformed from a numerical variable to a factor variable and observations were classified as either EADS or non-EADS. Observations with a class value of 0 were classified as non-EADs. Observations with a class value above 0 were classified as EADs. Figure 1 and table 2 show that only 2,332 (1%) of our observations are of the EAD class, meaning that the dataset is severely imbalanced.

Class variable	0	0.5	1	1.5	2
Number of occurrences	209917	635	1421	245	31
Proportion in dataset	98.90%	0.30%	0.67%	0.12%	0.01%

Table 2: Average number of EADs present in a 2-beat window

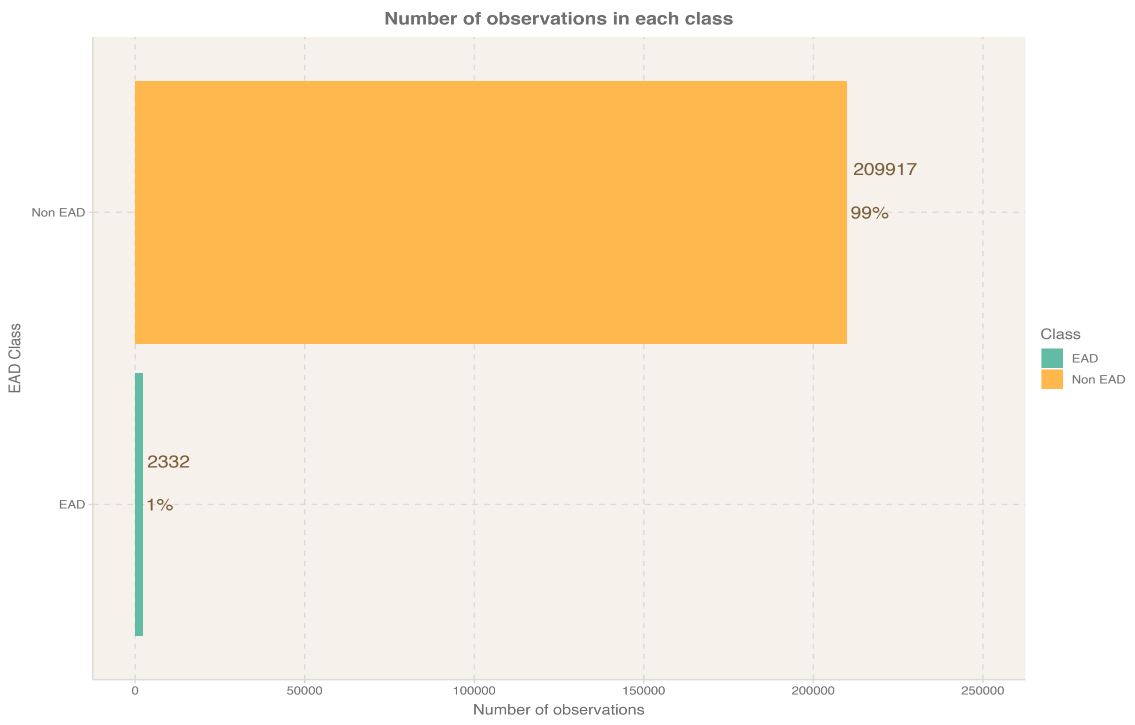


Figure 1: Number of observations in the dataset grouped by the Class variable

METHODOLOGY

All analyses for this project were done using RStudio 3.6.2. Four machine learning algorithms were compared in order to determine the best algorithm for classifying the observations as either EADs or 'Non-EADS' (normal action potentials). The algorithms tested were multivariable logistic regression, decision trees, random forest and extreme gradient boosting (XGBoost). Multivariable logistic regression and decision trees were considered because the rules and logic they use to make their predictions are readily available for us to view. Random Forests and XGBoost are more powerful algorithms but they are 'black-box' algorithms. As black box algorithms, although we can view their inputs and their outputs (such as the predictions they make), the inner rules and logic they use to turn the inputs into outputs are not readily available to us.

For this project, we are not just interested in classifying EADs and non-EADS. We aim to understand the parameter importance in making these classifications, therefore it is important for us to understand the rules and logic behind these decisions. For black-box algorithms, we compensated for this lack of visibility by using SHAP values to understand parameter importance.

Overview of the process

- The dataset was split into a training set and a testing set
- Models were trained using the training dataset. For each classification algorithm, multiple models were fitted on the training set to test different balancing techniques
- Predictions were made on the testing dataset to evaluate each model from each algorithm – insight into how the model performs on data it has never seen
- Evaluation metrics from a confusion matrix e.g. sensitivity, specificity and kappa, were used to determine the best performing models
- The best performing model from each classification method was chosen and they were then compared to one another (with evaluation metrics) to determine the best overall model
- SHAP (Shapley Additive exPlanations) values were used to determine feature importance, i.e. the features that were deemed by the model as most important to classifying observations from the dataset as EADs or non-EADs. Using these findings, we can gain insight into the ionic mechanisms that are most responsible for determining whether an action potential will be normal (non-EAD) or an EAD

Splitting the data and exploratory analysis

The dataset was split into a training set (75% of the dataset) and a testing set (25% of the dataset) using random sampling. The training set was used to train the model and the testing set was used to evaluate model performance. The training set contained 159,187 observations of which 1,749 were EADs and 157,438 were non-EAD. The testing set contained 53,062 observations. 583 were EADs and 52,479 were non-EAD observations. Both the training and the testing set had the same distribution as the full dataset.

Before the models were built, exploratory analysis was conducted on the training set (rather than the entire dataset). A summary table and scatterplot matrix were produced to summarize, visualize and essentially learn about the training data before the models were fitted. Any early insights and decisions made concerning the model should only involve the training set to lower bias and the chance of overfitting. Since the testing set is only used to evaluate the performance of the model, at this stage, we essentially pretend that the test set does not exist.

The summary table (Table 3) and scatterplot matrix (Figure 2) revealed interesting insights about the training data. Of all the 212, 249 observations, EADs were not present when scale factors of P_{Ca} were below 0.4499. Additionally, if scale factors of G_{Kr} were above 0.3358, EADs were not present. This is visualized in the scatterplots, density plots and boxplots where we do not see the presence of EADs beyond these points. Interactions without G_{Kr} and P_{Ca} generally show EADs scattered throughout the parameter space. However, scatterplots also indicate interactions between G_{Kr} and P_{Ca} , G_{Ks} and G_{Kr}, G_{NCX} and P_{Ca} and G_{NCX} and G_{Kr}. More specifically, areas with low G_{Kr} and high P_{Ca} had high EAD density, low G_{Kr} and G_{Ks} appear to have high density of EADs as well. High levels of P_{Ca} and G_{NCX} also show the high density of EADs and EAD density is also high when G_{Kr} is low and G_{NCX} is high. APD90 also seem to separate EAD and Non-EAD classes very well, showing agreement with previous research that prolonged APD90 can aid in EAD genesis [3]. APD90 was not explored beyond this exploratory analysis due to time constraints.

		Summary Statistics by EAD class					
Variable	**Class**	**Min**	**Q1**	**Median**	**Mean**	**Q3**	**Max**
G_{Na}	**EAD**	0.1001	0.5776	1.0673	1.0287	1.4885	1.9000
	Non EAD	0.1000	0.5545	1.0016	1.0016	1.4477	1.9000
G_{to}	**EAD**	0.1004	0.5308	0.9801	0.9818	1.4283	1.8961
	Non EAD	0.1000	0.5485	0.9928	0.9970	1.4461	1.9000
***P_{Ca}**	**EAD**	0.4499	1.2899	1.5441	1.4920	1.7418	1.8993
	Non EAD	0.1000	0.5465	0.9899	0.9929	1.4397	1.9000
***G_{Kr}**	**EAD**	0.1002	0.1274	0.1564	0.1646	0.1924	0.3358
	Non EAD	0.1000	0.5783	1.0200	1.0189	1.4611	1.9000
G_{Ks}	**EAD**	0.1009	0.4989	0.8624	0.8978	1.2526	1.8931
	Non EAD	0.1000	0.5574	1.0097	1.0052	1.4543	1.9000
G_{K1}	**EAD**	0.1004	0.4881	0.9664	0.9614	1.3944	1.8990
	Non EAD	0.1000	0.5523	1.0085	1.0041	1.4537	1.9000
G_{NCX}	**EAD**	0.1035	0.8918	1.3258	1.2223	1.6318	1.8986
	Non EAD	0.1000	0.5483	0.9896	0.9934	1.4398	1.9000
G_{pCa}	**EAD**	0.1033	0.5990	1.0200	1.0228	1.4753	1.8999
	Non EAD	0.1000	0.5498	1.0025	1.0014	1.4503	1.9000
***APD90**	**EAD**	67.74	1069.94	1180.34	1167.25	1300.73	1983.30
	Non EAD	101.80	222.50	286.0	327.20	396.60	1485.0

Table 3: Summary statistics for each EAD class. Variables with the most significant differences between classes are marked with ''*

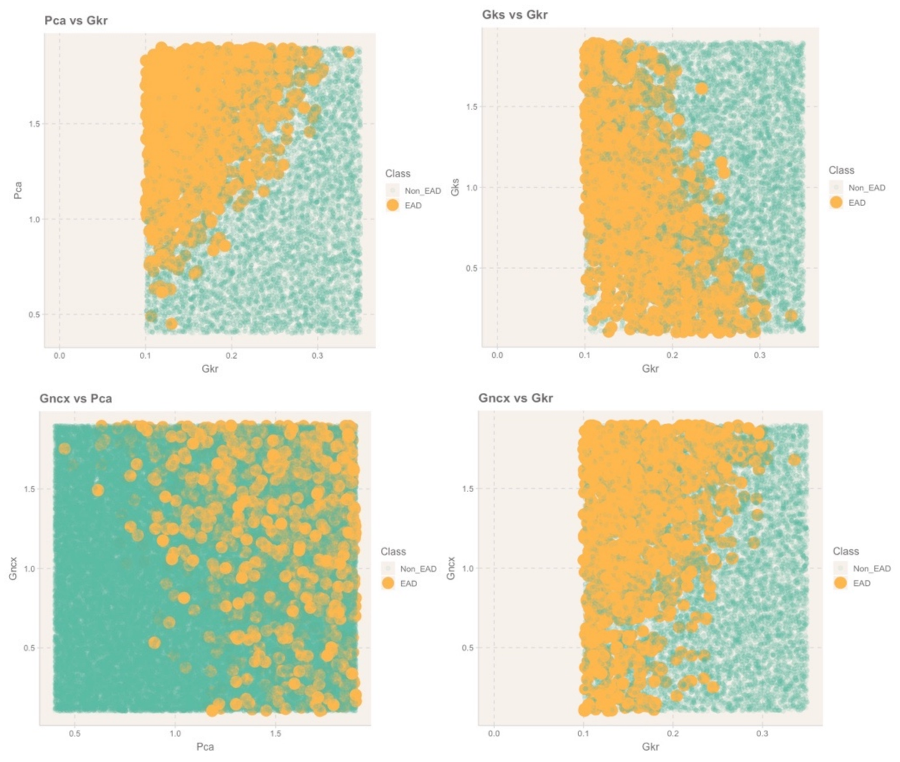


Figure 2A: Scatterplots showing key interactions between parameters

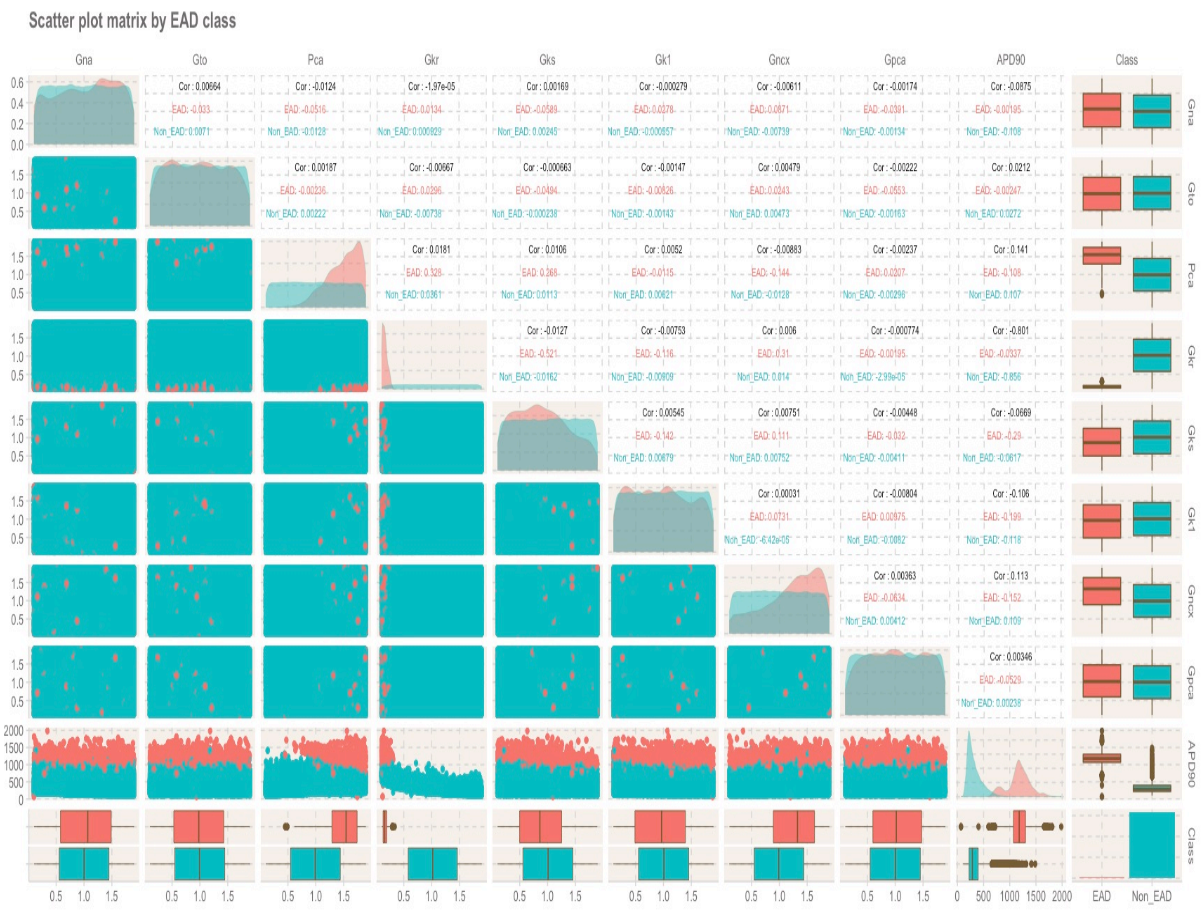


Figure 2B: Scatterplot matrix for the training dataset

Addressing class imbalance

Class imbalance can technically be used to describe datasets without a perfect 50:50 split between classes and a perfectly balanced dataset is not necessarily needed when building a model. Class imbalance becomes an issue when we have more significant between-class imbalance [15] - the proportion of classes can be 100:1, 1,000:1, 10,000:1, etc. In these scenarios, one of the classes is heavily underrepresented and it's quite common in the biomedical domain where we are often interested in identifying and understanding rare events. Classifying using imbalanced data is an issue because many machine learning algorithms are designed with the assumption of low-dimensionality, equal misclassification costs and class distributions being balanced. With this in mind, these algorithms are therefore designed to prioritise overall accuracy and for datasets such as the one used in this project, accuracy above 95% can easily be achieved by simply classifying every observation to the majority class (as a non-EAD).

We wanted to produce a classifier that punishes false positives and false negatives equally and with high accuracy. There are a few methods that we can utilise to account for class imbalance and three such methods were used in this study: using class weights or sampling methods during training and using appropriate evaluation metrics as part of cross validation and/or during model evaluation. Class weights and sampling methods are described directly below and appropriate evaluation metrics are discussed later in the report under *Model Evaluation* in the **Results** section.

Class weights – can be used to balance the dataset according to class distributions. They dictate the extent to which false positives and false negatives are punished. We are interested in finding the best performing model that has the best balance in correctly classifying EADs and Non-EADs. Therefore, class weights were set to equally punish false positives and false negatives so that the classifier will learn from both classes equally.

Data sampling – balances the distribution of observations in the minority and majority classes of the training set so that the minority class is not overlooked by the classifier [16] For this project, four sampling methods were considered and tested on each of the algorithms : random oversampling, random undersampling, ROSE (random oversampling examples) and SMOTE (synthetic minority oversampling technique).

- Random oversampling - randomly selects observations of the minority class (EADs), replicates them (can be done more than once) and adds them to the training set [15] . Enough samples of the minority class are generated to equal the number of observations in the majority class. This sampling method may lead to overfitting.

- Random undersampling – randomly selects and removes observations of the majority class (non-EADs) so that the number of observations of the majority class match the number of observations of the minority class [15]. An obvious drawback of this sampling method is the potential of discarding important data leading to the classifier missing useful information regarding the majority class.
- ROSE – removes samples from the majority class and synthesizes new samples of the minority class using a smooth bootstrap approach. Synthetic observations are generated using a conditional density estimate of both classes [17].
- SMOTE [18] – also removes samples from the majority class and synthesizes new samples from the minority class but it uses a k-nearest neighbours and bootstrapping approach. SMOTE focuses on the parameter space rather than the data space. To create synthetic samples, an observation in the parameter space and its k nearest neighbours are selected at random. We take the difference between the observation and one of the k neighbours and multiply the difference between a random number between 0 and 1. This creates a new observation that is a linear combination of those two observations and this new observation is added to the parameter space. One of SMOTE's drawbacks is that it doesn't recognize that neighbouring observations can be from a different class and can therefore introduce noise.

Sampling was only done on the training set and the testing set remained untouched. For multivariable logistic regression and XGBoost models, new training sets were generated using a sampling function through the ROSE package (for undersampling, oversampling and ROSE) and DMwR package (for SMOTE sampling) [17][18]. For decision trees and random forests, the Caret package [19] was used for training and it allowed sampling to be implemented as part of the cross-validation procedure.

Multivariable Logistic Regression

Theory

Multivariable logistic regression is a special case of regression where the response variable Y is a binary/categorial variable and we have multiple predictor variables X. For this study, we have eight predictor variables and the logistic regression model can be written as:

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_8 x_8}$$

Taking the log of both sides:

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_8 x_8$$

It is the probability of an event being true divided by the probability of the event not being true. It is also referred to as the logit function. Solving for p, we arrive at the formula below

$$p = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_8 x_8}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_8 x_8}}$$

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_8 x_8)}}$$

In the context of this classification problem, p is the probability that the observation is an EAD and $1 - p$ is the probability that the observation is a non-EAD. The coefficients (βs) are estimated by Maximum Likelihood Estimation (MLE). β_0 is the intercept and $\beta_1 \dots \beta_8$ are the regression coefficients associated to each predictor variable $x_1 \dots x_8$. The βs denote the effect that changes in each parameter have on the probability that a certain outcome will occur.

Unlike most programming languages, indexing in R begins at one instead of zero. For this analysis, our positive class 'EAD' is defined by R as $Y = 1$ and our negative class 'non-EAD' is defined as $Y = 2$. A positive β indicates that an increase in its associated parameter would increase the probability that an observation is a non-EAD (1-p) and a negative β indicates that an increase in its associated parameter would increase the probability that an observation is an EAD (p). Additionally, the magnitude of the βs will show which parameters are deemed the most important by the model, i.e. the parameter with the largest coefficient (in either the negative or positive direction) is the most important parameter in making the prediction and the parameter with the smallest (in either the negative or positive direction) is the least important.

Model Fitting

A saturated logistic regression model (with all 8 predictor variables) was fitted to the training dataset which was imbalanced. Saturated logistic regression models were also fitted to the sampled datasets: oversampled training dataset, undersampled training dataset, ROSE sampled training dataset and the SMOTE sampled dataset. Predictor variables were not removed from the model because we aim to understand the importance of all of the predictor variables.

These 5 models were fitted using the *glm* function. The family argument was set to binomial with a logit link in order to fit a logistic regression model rather than another generalised linear model. Predictions on the 5 training datasets were made from each of the models giving 5 vectors of fitted probabilities. Each vector corresponded to a trained model. Fitted probabilities from the training set were used to find optimal prediction probability threshold values (p^*). These p^*s were found by plotting the sensitivity and specificity from the fitted probabilities on the y axis and threshold values on the x axis (Figure 3). Thresholds that had the best trade-off between sensitivity and specificity were chosen for each of the 5 models. The optimal p^* according to Figure 3 were 0.97 for the imbalanced training model, 0.5 for the undersampled, oversampled and SMOTE models and 0.25 for the ROSE model.

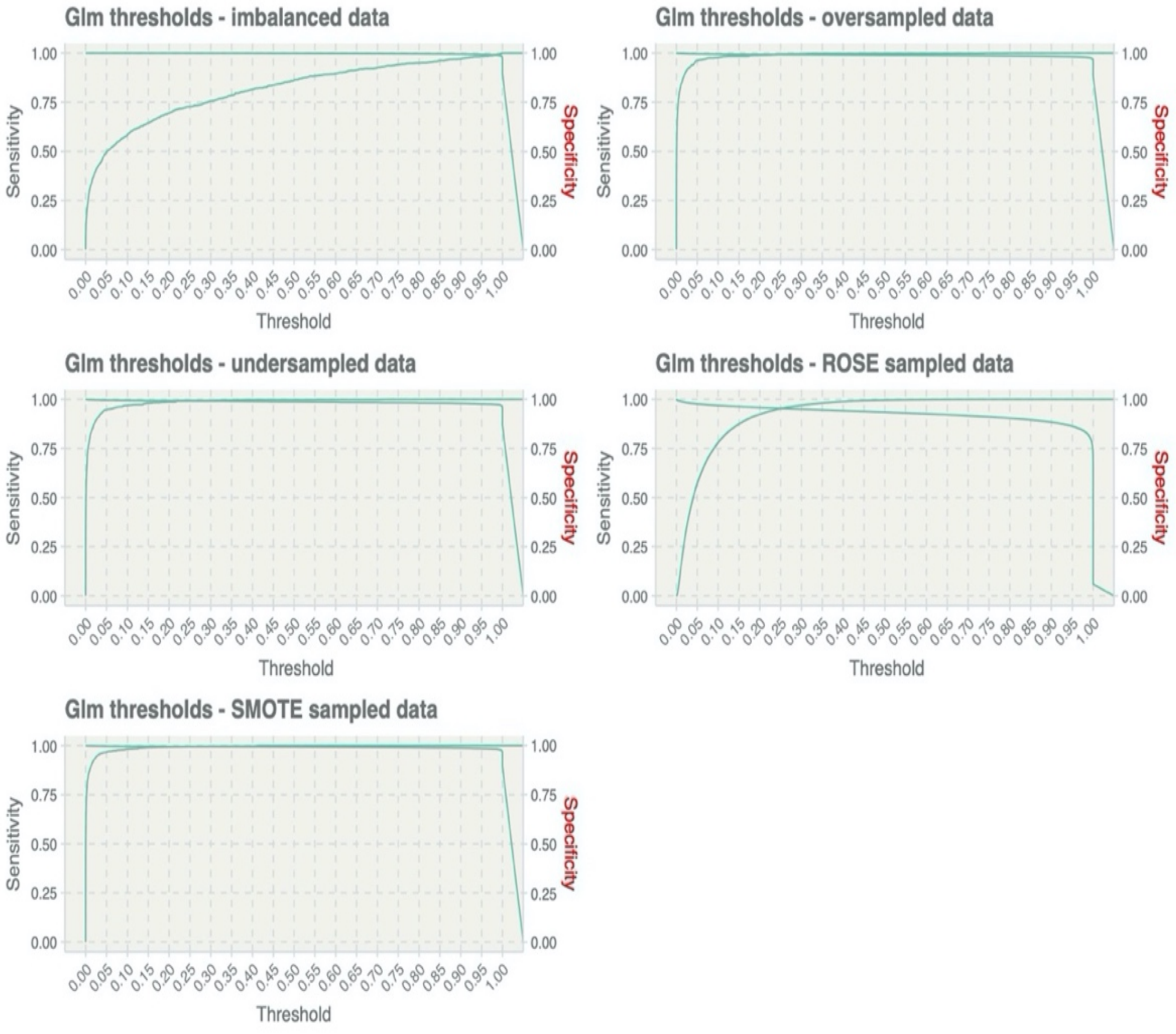


Figure 3: Plots showing sensitivity-specificity trade-off for the multivariable logistic regression models at corresponding p^* values.

Predictions were then made on the testing set to obtain 5 new vectors of fitted probabilities using the trained models. If a fitted probability is above p^*, a non-EAD is predicted. If a fitted probability is below p^*, an EAD is predicted.

Confusion matrices for each logistic regression model were computed. Each confusion matrix compared the predicted outcomes to the actual outcomes.

Decision Trees

Theory

CART models (Classification and regression trees) which can also be referred to as decision trees, can be used to solve classification problems through recursive partitioning of a predictor space in order to predict categorical variables such as EADs vs non-EADs. The predictor space is populated with all possible values of the input parameters/predictors, i.e. $X_1 \ldots X_d$. Decision trees repeatedly divide the predictor space into distinct subset regions, i.e. $R_1 \ldots\ldots R_k$, based on splitting variables j and split points s until a certain stopping rule is met. Where $x = (x_1, \ldots x_d) \in R^d$, $j \in \{1,2 \ldots d\}$ and $s \in R$, distinct subregions 1 and 2 can be shown as:

$$R_1(j,s) = \{x | x_j \ \leq s\}$$
$$R_2(j,s) = \{x | x_j \ \geq \ s\}$$

The aim is to predict the most common class among the data within each region and the stopping criterion could be that:

- All of the data in a region are of the same class
- The set number of partitions have been met - the size limit is something the programmer can control
- There are no remaining parameter values or features that the algorithm can partition on

This can also be explained with a tree structure. Each observation under consideration corresponds to a specific set of feature values which can either be categorial or numerical. For a classification issue, the classifier will use these parameter values to predict the class that each observation belongs to. At the top of the tree, the entire dataset is represented because no splitting has occurred as yet. An observation first passes through a root node which is the best predictor (and first decision node). The observation either meets or doesn't meet ('yes' or 'no') this parameter criteria. Based on the outcome of that decision node, it can then be passed to other decision nodes along branches until a final decision is made regarding an observation. When that final decision has been made, the decision tree terminates with terminal nodes which specify the expected class that the observation belongs to, given the decisions made at previous decision nodes in the tree. This tree like structure is one of the benefits of decision trees as the output can be visualized and shows how the model makes its decisions and ranks the parameters it considers important when making these decisions.

More specifically, if a dataset contains class values of $1 \ldots k$ and N observations of the form (x_i, y_i) for $i = 1,2,3 \ldots N$, if a node m represents an R_m containing N_m observations [20]:

- The proportion of observations belonging to class k in m will be

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i: x_i \in R_m\}} I(y_i = k)$$

- Observations in node m are classified to the majority class in that node using the function
$$k(m) = argmax_k \hat{p}_{mk}$$

Ideally, we want terminal nodes that are pure (containing observations of a single class) but this is not always the case. Therefore, we need measures of node impurity $Q_m(T)$, three of which are the misclassification error, cross-entropy and deviance [20]:

- Misclassification Error : $\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i: x_i \epsilon R_m\}} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$
- Cross-entropy: $- \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$
- Gini index : $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk} (1 - \hat{p}_{mk})$

These measures are also shown in Figure 4 [20]. In the analysis done in this report, the *rpart* function from the Caret package [19] uses Gini index as its $Q_m(T)$. For our two-class classification problem, if p denotes the proportion in the second class, the Gini index will be $2p(1-p)$.

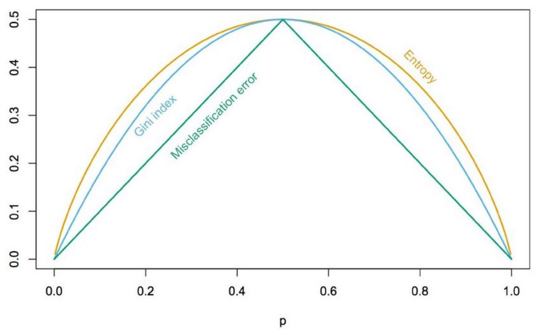


Figure 4: Measures of node impurity for two class classification
Source: Hastie et al. 2008 [20, Fig 9.3]

Decision Trees (classification trees) is a popular classification method partly due to the fact that the output is a logical tree structure that doesn't require a math background to interpret. Hence, their output is usually easily understood by all stakeholders. They require minimal pre-processing and the data does not need to be scaled or normalised before training the model. Missing data and outliers have minimum impact when training the model and it can accommodate numeric and categorical features. Small and large datasets can be used in the model and it automatically performs variable selection, removing variables that the model thinks are unimportant.

Classification trees also have some weaknesses. Although the output is rather intuitive and the decisions are easily understood, as the trees become larger the output becomes more difficult to interpret. Classification trees also use axis-parallel splits. This can be seen as a disadvantage because in some classification problems, a diagonal split could provide a cleaner partition but due to classification trees' lack of flexibility in this regard, partitioning the parameter space with more complex decision boundaries are not possible.

Running the models can also be computationally expensive and they are susceptible to both overfitting and underfitting. Classification trees will grow indefinitely, partitioning the parameter space until each observation is perfectly classified or it runs out of parameters to split on or one of the other stopping criteria (mentioned above) is met. This will lead to overfitting on the training data. To prevent or reduce overfitting, the trees can be pruned. By pre-pruning the tree, we can dictate how many trees should be grown or impose a penalty for too many splits that do not significantly improve the performance of the model. However, by pre-pruning we may stop the tree too early and miss insightful and important patterns/splits, so it leads to underfitting. Post-pruning is also an option. In this case, a large tree that overfits the data is grown, once this is done, nodes and branches that have little effect on classification errors are pruned. This pruning method has the benefit of discovering as many important patterns/splits as possible before being pruned, however, over-pruning in this way can lead to underfitting and getting rid of important information. Therefore, the balance between overfitting and underfitting can be quite challenging.

Another disadvantage to classification trees is their instability and susceptibility to high variance. Small changes in the training data can have a significant impact on the decisions of the model and the overall structure of the tree. For instance, trees are grown in a hierarchical manner so a decision error made near the top of the tree will negatively impact the other nodes and decisions that follow it. There are different ways to approach this high variance issue including ensemble-based algorithms like random forests and XGBoost.

Model Fitting

Six decision tree models were fitted to the training data using the Caret package [19]. All models were trained using the original training set. For the model without weights or data sampling, resampling was done through repeated K-fold cross-validation (repeated-CV) using area under the ROC as the metric.

An ROC curve is a plot that shows the trade-off between Sensitivity and Specificity as the decision threshold values vary [21]. It plots Sensitivity on the y axis and 1- Specificity on the x axis. Low decision threshold values closer to point 1,1 will have higher sensitivity and lower specificity. High decision threshold values closer to point 0,0 will have lower sensitivity and higher specificity. The area under the AUC plot is a single value that shows how close the ROC curve is to being perfect [22]. AUC values range between 0.5 and 1. An AUC value of 0.5 means that there is no distributional difference between observations of the two classes while an AUC value of 1 indicates that the model perfectly separates observations of the two classes [22]. AUC was used as the metric during training instead of accuracy to ensure a better balance between sensitivity and specificity.

Cross-validation divided the dataset into k Folds after shuffling the data at random. The decision tree model was trained using k = 10. One partition of 10 is 'held out' as a test set that the model is evaluated on and the rest are treated as the training set that is used to fit the model. By using repeated cross-validation where the number of repeats was chosen as 5, cross-validation was done five times for a decision tree model. The repeated-CV technique used, estimated the average AUC that we should expect on new unseen data for each complexity parameter.

The area under the ROC Curve (AUC) was used as a metric during repeated-CV to find the model corresponding to the optimal complexity parameter (cp) and highest AUC. The cp will impose a penalty if the trees have too many splits (pruning the tree). The optimal cp is the one that corresponded to the highest cross-validation AUC. Lower cps result in larger trees and higher cps result in smaller trees.

Another model with weights was trained using the same analysis above but with a weight argument that was added during training to assign equal weights to both classes.

4 additional models were trained using the same analysis described above but instead of using weights, data sampling was added as a part of the cross-validation process. Therefore, additional training sets did not need to be generated.

Predictions were then made on the testing set.

Random Forest

Theory

Random Forests [23], which is an ensemble-based method, combines the principles of feature selection and bagging (bootstrap aggregation). By randomly selecting input features and decorrelating trees through bagging, random forests can reduce the high variance associated with decision trees [20]. Utilising ensembles of decision trees to make predictions, random forests will 'grow' many trees and for a classification problem, each tree will perform classification, 'casting' its vote for the predicted class. The model will then choose the predicted class with the most votes. Again, because the trees' predictions are combined, random forests can reduce the high variance associated with decision trees. [24].

The random forest classification algorithm works as such [20]:

- For $b = 1$ to B, a bootstrap sample (Z^*) of size $N$is drawn from the training set.
- A random forest tree T_b is grown from this bootstrapped sample and for each terminal node the following steps are followed until the minimum node size of n_m is reached:
    - m parameters are selected at random from p parameters

    - Among the m parameters, the best variable/split is selected
    - A decision node is split into two further nodes
- The ensemble of trees produce the output : $\{T_b\}_1^B$
- For classification, in order to make a prediction at a new point x,

$$\hat{C}_{rf}^{B}(x) = majority\ vote\ \{\widehat{C_b}(x)\}_1^B$$

  where $\widehat{C_b}(x)$ is the class prediction of the bth random forest

In addition to all the pros for classification decision trees (except interpretability), they are less prone to overfitting. Random forests, due to bagging, can generalize over the data because the input variables are chosen randomly during training, unlike classification trees that give high importance to a particular set of input variables. This can lead to a more accurate classifier. They can also handle very large, high-dimensional datasets because the ensembles use small random selections of the full parameter sets to make decisions.

Random forests do have some cons as well. They can have a much higher training time than decision trees and as the number of trees grown increases, the longer the training time and computational power. The biggest drawback to random forests is that unlike decision trees, the model is not easily interpretable but there are ways to interpret a random forest output, such as with SHAP values.

Model fitting

Six random forest models were fitted to the training data using the Caret package [19] and the training process was the same as for the decision tree models with a few changes.

10-fold cross-validation was used instead of repeated 10-fold CV because R crashed multiple times when repeated-CV was attempted.

The mtry parameter was tuned instead of the cp parameter. Mtry dictates the number of parameter variables from the full set of parameter variables that were randomly sampled at each split. The area under the ROC Curve (AUC) was used to find the model corresponding to the optimal mtry and highest AUC.

XGBoost

Theory

XGBoost (Extreme Gradient Boosting) is a powerful, scalable, tree boosting algorithm. As a gradient boosting algorithm, it converts weak learners into strong ones and errors are minimized by a gradient descent algorithm. Trees are added sequentially and iteratively;

previous errors in a tree are resolved by the new trees that are built. XGBoost started as a research project by Tianqi Chen and was open sourced in 2014. Since its release, it's been recognized as a state-of-the-art machine learning algorithm highly favoured by data scientists and it's an algorithm that has been used in winning solutions for numerous challenges. In 2016, Tianqi Chen and Carlos Guestrin published their work in a paper entitled *XGBoost: A Scalable Tree Boosting System* [25]. The authors explain that the algorithm's success is due to important system and algorithmic optimisations [25]:

- Parallelization and distributed computing make learning faster, enabling quick model exploration
- Ability to handle sparse data
- Core computation through a cache aware block structure enables hardware optimization
- Cross validation is built into the model
- Penalizing complex models through regularization in order to prevent overfitting of the model

The XGBoost algorithm described by Chen and Guestrin [25]:

We have a dataset with n examples and m features (our parameters). Consider that x is an input vector, T represent the number of leaves in a tree and q is an independent tree structure with leaf weights w. The output of a single tree is $f(x) = w_q(x_i)$. The ensemble of trees is denoted as:

$$\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \longrightarrow T, w \in \mathbb{R}^T)$$

K additive functions are used to predict the output of a tree ensemble model:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

The model minimizes a regularized objective which is made up of a train loss function l (L1) and a regularization term Ω (L2). Where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2$, the regularized objective term is given as:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

The loss function measures the difference between the predicted $\hat{y}_i$ and the actual y_i and the regularization term helps the model to avoid overfitting by penalizing the complexity of the model.

The model is trained in an additive manner such that when $\hat{y}_i^t$ is the prediction of the i-th instance of the t-th, the f_t that is the best at improving our model is added greedily to minimize this objective :

$$\mathcal{L}^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

Using 1st and 2nd order gradient statistics which are

$$g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i , \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

and

$$h_i = \partial^2{}_{\hat{y}^{(t-1)}} l\left(y_i , \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

respectively, second order approximation of the loss function will find the optimal value of the objective function that corresponds to the optimal weight w_j^* of leaf j for a fixed structure $q(x)$. Where $w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$, the optimal value of the objective function is given as

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$$

Model fitting

Similar to the multivariable logistic regression models, 6 XGBoost models were fitted using the original training set (to fit one model without weights or sampling and a model with weights and without sampling) and the four data sampled training sets.

Before the models were fitted, the training sets were converted into a format that the XGBoost algorithm would accept. The training sets which were originally data frames were coerced into matrices. The class values were also converted to character variables and EADs were re-labelled as 0 and non-EADs were re-labelled as 1.

The default classification arguments tree booster (*gbtree*) and *binary: logistic* were used to train the model. Most of the default parameters were used, however, using XGBoost's built-in cross validation, 10-fold CV was used to find the optimal number of iterations and test-auc was used as the metric. XGBoost is a 'greedy' algorithm. Instead of thinking about the best choice for the overall model, a greedy model makes the decision that optimizes the objective function at each step. To prevent overfitting, as part of cross-validation, early stopping rounds of 5 were used. By setting this criterion, cross-validation stopped when the test-auc did not improve in 5 iterations.

The optimal number of rounds, found through this cross-validation step, was then used to build an XGBoost model. This was done for all 6 XGBoost models.

Predictions were made using the test set. XGBoost only performs regression, therefore the outputs of the predictions are predicted probabilities. To convert these into classifications, if the predicted probability was less than 0.5 an observation was classified as 0 (EADs) and if the predicted probability was above 0.5, they were classified as 1 (non-EADs).

Model Evaluation

A confusion matrix, shown in Table 4 is a great way to visualize the classification results of a model. Once predictions were made, confusion matrices for each model were computed. Each confusion matrix compared the predicted classifications to the actual classifications.

Using those comparisons, we can calculate certain statistical measures such as accuracy, sensitivity and specificity and use them to assess the performance of the model.

		PREDICTED CLASS	
		Positive Class	**Negative Class**
ACTUAL CLASS	**Positive Class**	True Positive (TP)	False Negative (FN)
	Negative Class	False Positive (FP)	True Negative (TN)

Table 4: Example of a confusion matrix

The most used performance metrics are accuracy (proportion of correct predictions) and the error rate (proportion of incorrect predictions).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Accuracy = \frac{True\ EADs + True\ Non\ EADs}{True\ EADs + True\ Non\ EADs + False\ EADs + False\ Non\ EADs}$$

These are not appropriate metrics when we have imbalanced data. When a classifier trains on heavily imbalanced data, it can easily achieve almost perfect accuracy by simply classifying all observations as belonging to the majority class. This can often come with a big cost. For example, if all EADs are misclassified as non-EADs, we will not be able to understand key channels and currents responsible for generating EADs that can lead to fatal arrhythmias - making the classifier essentially useless for our purposes.

Ideally, we would like the algorithms to prioritise the predictive accuracy of both the minority and majority classes, but these algorithms by design, are programmed to prioritise the overall accuracy. To determine the best classifier, sensitivity, specificity, precision (positive predictive value), negative predictive value, balanced accuracy, F1 and kappa values were all taken into consideration. These are more appropriate metrics for datasets with a class imbalance problem. They will be used in this analysis to evaluate model performance. Where EADs are the positive class and non-EADs are the negative class:

- Sensitivity – also known as recall or the true positive rate, is the proportion of genuinely positive observations that were classified as positives. This metric will indicate the proportion of true EADs that were classified as EADs

$$Sensitivity = \frac{TP}{TP + FN} = \frac{True\ EADs}{True\ EADs + False\ Non\ EADs}$$

- Specificity – also known as the true negative rate or selectivity, is the proportion of genuinely negative observations that were classified as negatives. It represents the proportion of true non-EADs that were classified as non-EADs

$$Specificity = \frac{TN}{TN + FP} = \frac{True\ Non\ EADs}{True\ Non\ EADs + False\ EADs}$$

- Precision – also known as the positive predictive value, shows proportion of observations that were classified as positive that are actually positive. This measure will indicate the proportion of classified EADs that are genuine EADS

$$Precision\ = \frac{TP}{TP + FP} = \frac{True\ EADs}{True\ EADs + False\ EADs}$$

- Negative predictive value – shows proportion of observations that were classified as negative that are actually negative. This measure will indicate the proportion of classified non-EADs that are genuine non-EADS

$$Negative\ predicitve\ value\ = \frac{TN}{TN + FN}$$

$$Negative\ predictive\ value = \frac{True\ Non\ EADs}{True\ Non\ EADs + False\ Non\ EADs}$$

- Balanced Accuracy – is the mean of the sensitivity and specificity i.e. the mean of the proportion of correctly classified observations

$$Balanced\ Accuracy\ = \frac{Sensitivity + Specificity}{2}$$

- F1 score – is commonly referred to as the 'harmonic mean' of precision and recall [26]. The F1 score is helpful when we have imbalanced data and we are equally interested in having great precision and recall (such as in this study). The formula is defined as such [26]

$$F_1 = \frac{2}{\frac{1}{Precision} \times \frac{1}{Recall}} = \frac{2\ (Precision\ \times\ Recall)}{Precision + Recall}$$

$$F_1 = \frac{TP}{\frac{TP + (FP + FN)}{2}} = \frac{True\ EADs}{\frac{True\ EADs + (False\ EADs + False\ Non\ EADs)}{2}}$$

- Kappa – formally known as Cohen’s kappa statistic, measures how well a classifier performs in comparison to chance. A higher kappa indicates that there is a more significant difference between the classifier’s performance and simply guessing/classifying the observations according to their class distribution. The accuracy measure has values ranging between 0-1. A kappa value of 0 means that we should disregard the classifier as it is as good as chance, 1 means that there is perfect agreement. We would like to have a kappa of at least 0.8. Values about this indicate that we have a great classifier. Cohen’s kappa, where P_o is total agreement probability due to accuracy and P_c is the agreement probability due to chance [27]:

$$K = \frac{P_0 - P_c}{1 - P_c}$$

SHAP values

Shapley values [28] is a concept taken from game theory. For a prediction of y ($\hat{y}$), the shapley value measures the contribution of the parameter in pushing an actual $\hat{y}$ away from the expected value of $\hat{y}$. If F represents all of our parameters and $S \subseteq F$ represents all our parameter subsets, for each parameter we will be able to assign a value which represents the parameter’s effect on a model’s prediction – an importance value [29]. In order to calculate this effect, consider that x_s are the values of the input parameters in a finite coalition S. Two models are trained: one with the feature present $f_{S\cup\{i\}}$ and one with the feature withheld f_S, then we compare predictions from both models on the current input $f_{S\cup\{i\}}\left(x_{S\cup\{i\}}\right) - f_S(x_s)$ [29]. As we can see, withholding a parameter will depend on other parameters in our model so preceding differences are calculated for all parameter subsets $S \subset F \setminus \{i\}$; the shapley values are used as parameter attributions and are calculated as [29]

$$\phi_i = \sum_{S \subseteq F\setminus\{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \left[f_{S\cup\{i\}}\left(x_{S\cup\{i\}}\right) - f_S(x_s)\right]$$

SHAP (Shapley Additive exPlanations) was presented by Lundberg and Lee [29]. It’s a special form of the shapley value such that if we have a model, SHAP values can explain the output of this model as the sum of the parameters’ shapley values of a conditional expectation function [29]. Each parameter is introduced one at a time into a conditional expectation function of the model’s output, the change produced at each step is attributed to the parameter that was introduced and this process is averaged over all possible orderings of the parameters [30]. Therefore, SHAP values show the impact each parameter has on the model output; SHAP can explain the prediction of an observation as the sum of the contribution of each individual parameter.

Lundberg et.al [31] have also introduced TreeSHAP which calculates exact SHAP for tree-based models. TreeSHAP computes local explanations based on exact shapley values in polynomial time rather than exponential time, leveraging the structure of tree-based models to render fast results [31].

In our project, we are interested in global parameter importance. An advantage of SHAP is that the shapley values can be combined forming a matrix of shapley values which will give us global explanations of the model. In this matrix, each column represents a parameter and each row represents an observation. We can use this matrix to understand and visualize the model's parameter importance when making predictions through SHAP plots.

SHAP values and plots were produced using the *SHAPforxgboost* package [32] which produces SHAP values from an XGBoost model to create SHAP plots in R. This package replicates the original *shap* [33] python package (not available in R) developed by Lundberg. Summary and dependence plots were used to understand parameter attribution and interactions in determining whether an observation was classified as an EAD or a non-EAD. Meaning that, we used these plots to understand how changes in class probability can be attributed in different proportions to the parameters of the model.

RESULTS

Model selection from each algorithm

Confusion matrices and the metrics produced from them were used to determine the best model from each algorithm and then the overall model. These can be found in Table 5 and Figure 5A-D. For all algorithms, the best performing model was the model without any sampling or weight arguments as shown by their kappa scores. At best, such as in the oversampled random forest models, their performance was only slightly less than that of the original model. For many of the sampled models however, they had kappa scores less than 0.5 and were therefore rather poor classifiers.

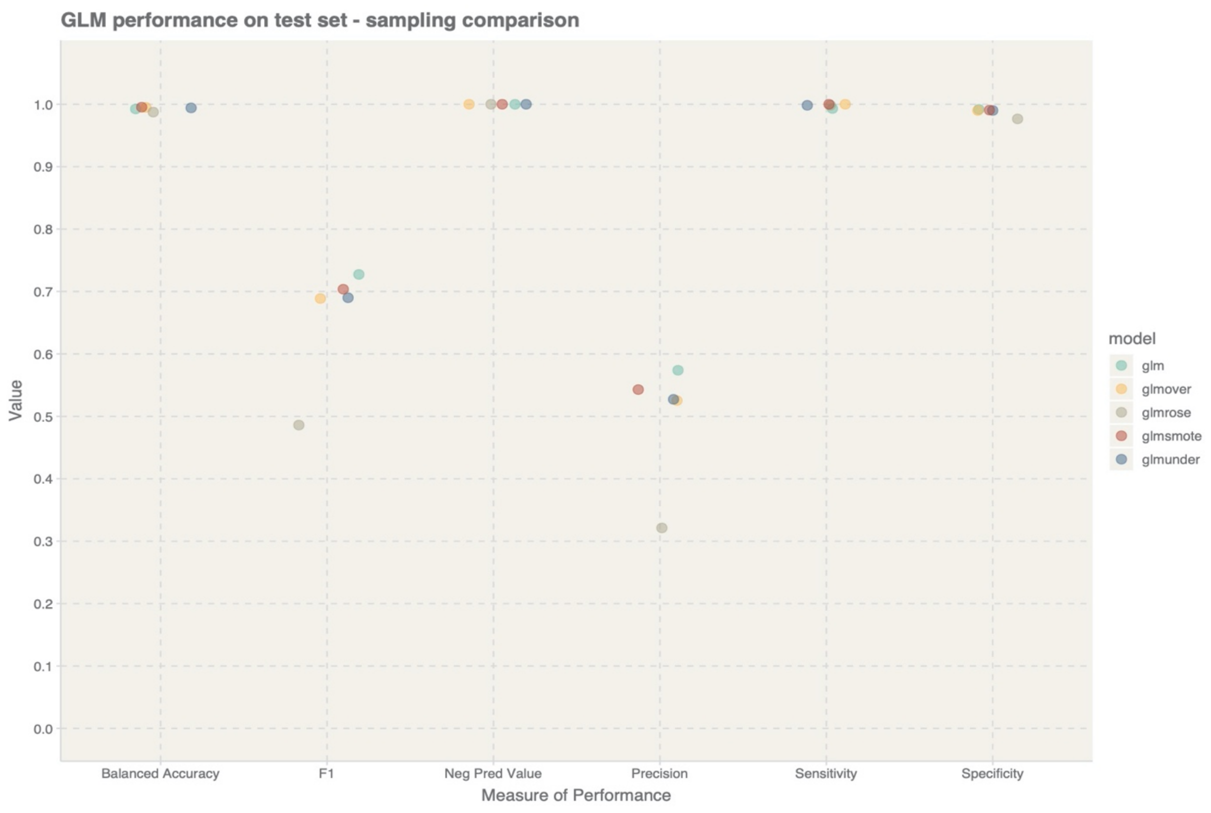
Data sampling techniques have been shown to be great techniques to correct class imbalance, [15] but this does not work for every dataset. One reason could be the fact that when undersampling (used in undersampling, Rose and SMOTE) we lose majority of the information from the dataset and when oversampling, we are synthesizing a significant amount of our dataset. However, what we see here might be a case of 'over-correcting'. We may have compensated for the imbalance too much and as a result created decision boundaries that do not reflect the true nature of the data. Hence, when applied to a new dataset such as our testing set which has severe imbalance, the new decision boundaries do not fit the new data well. Thankfully, EADs are a rare occurrence but this means that future datasets are likely to have imbalance which need to be accounted for.

For logistic regression, simply choosing the optimal threshold led to the best performing logistic regression model (although it was not an excellent classifier). For the tree-based models (decision trees, random trees and XGBoost) the best balance between low misclassification of True EADs and True normal APs (non-EADs) was found by simply using AUC as a metric during cross-validation. By cross-validating using this metric as opposed to high accuracy for example (which can be achieved by classifying everything as non-EADs) we are telling the algorithm to find the best balance between reducing false positives and false negatives.

Algorithm	Model	Kappa
Multivariable Logistic Regression	Original	0.724
	Undersampled	0.685
	Oversampled	0.684
	ROSE	0.477
	SMOTE	0.699
Decision Trees	Original	0.794
	Weights	0.452
	Undersampled	0.363
	Oversampled	0.447
	ROSE	0.301
	SMOTE	0.478
Random Forests	Original	0.935
	Weights	0.935
	Undersampled	0.425
	Oversampled	0.923
	ROSE	0.397
	SMOTE	0.607
XGBoost	Original	0.969
	Weights	0.834
	Undersampled	0.481
	Oversampled	0.876
	ROSE	0.410
	SMOTE	0.570

Table 5: Kappa values for each model

5A



5B

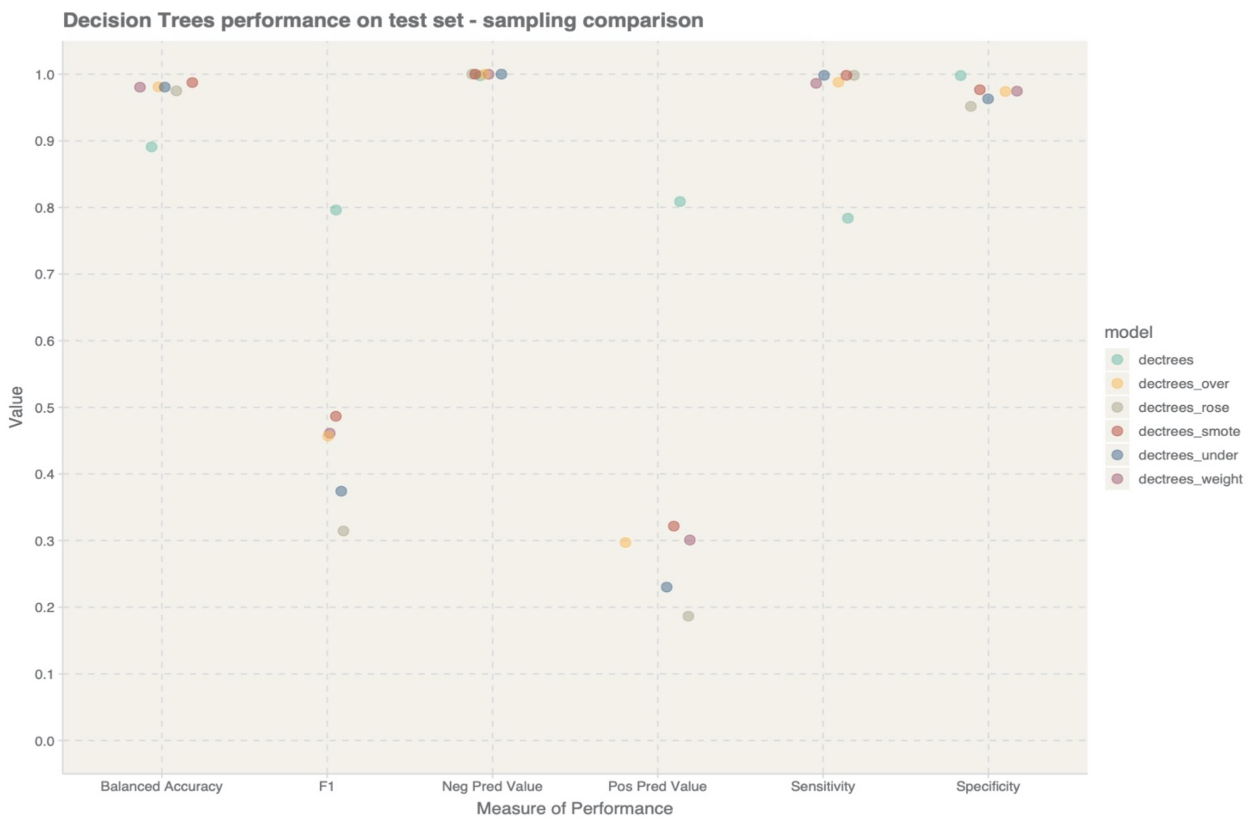
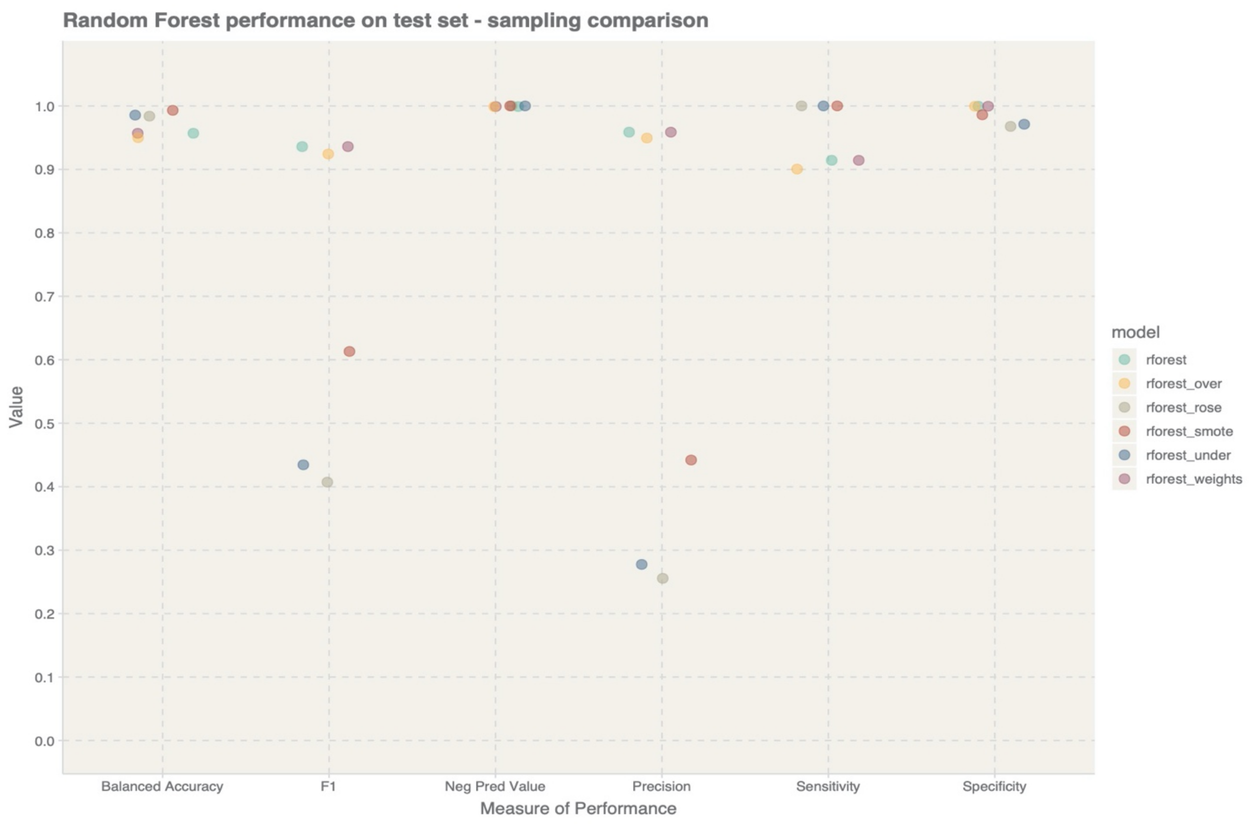


Figure 5(A-B): Figure 5A compares metrics (from confusion matrices) for all multivariable logistic regression models that were fitted. Figure 5B compares metrics (from confusion matrices) for all decision tree models that were fitted.

5C



5D

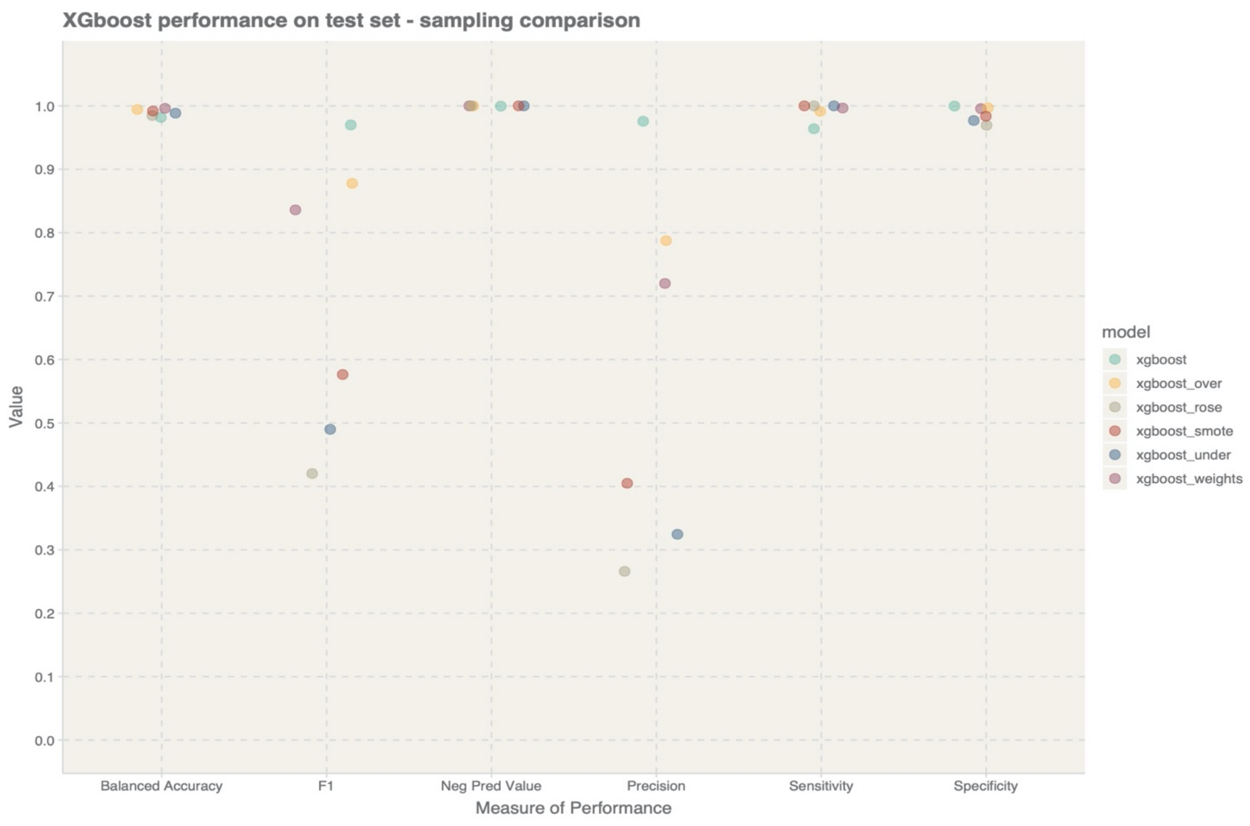
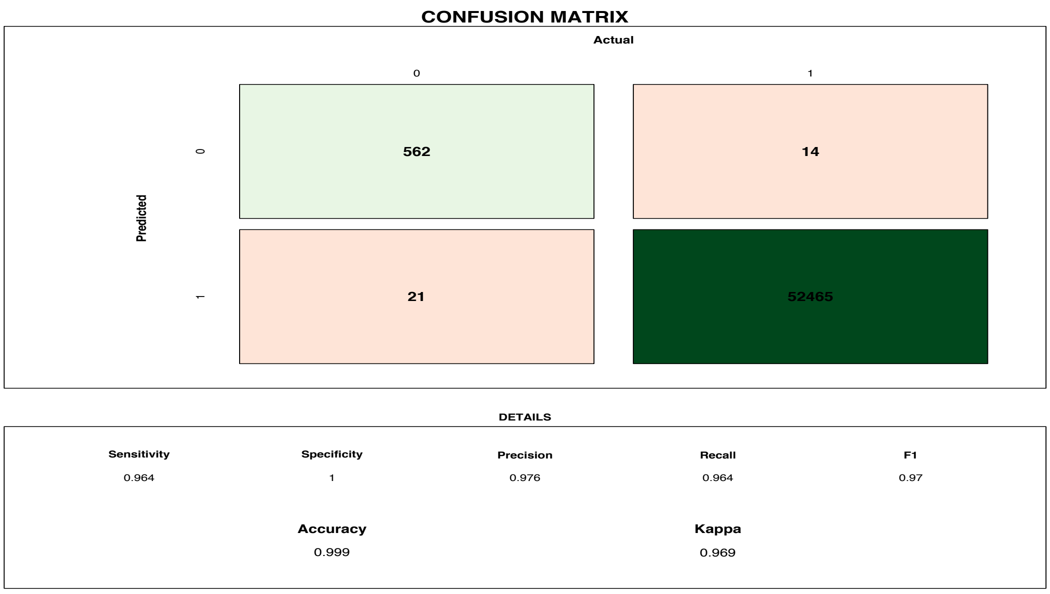


Figure 5(C-D): Figure 5C compares metrics (from confusion matrices) for all Random Forest models that were fitted. Figure 5D compares metrics (from confusion matrices) for all XGBoost models that were fitted

Best overall model

We were ideally hoping to train models with kappa scores above 0.8, showing that the model is a very good classifier. We also wanted there to be a good balance between sensitivity and specificity although we understand that there is a trade-off between the two. Additionally, we wanted to achieve good precision - this value was low for many of the models. For example, the logistic regression models mostly had approximately perfect sensitivity but they misclassified a large number of non-EADs. Relative to the number of non-EADs in the dataset, the number of misclassifications only represented a very small percentage of the non-EADs which is why specificity remained almost perfect. However, the presion metric gave a clear idea of when there were a lot of false EADs relative to the number of true EADs. Regardless, we were able to train an excellent classifier – XGBoost without weights or data sampling trained with number of rounds equal to 134.

As shown in Figures 6 and 7, with a kappa value of 0.969, XGBoost without weights or sampling (going forward it will simply be referred to as the XGBoost model) slightly outperformed the random forest model. The confusion matrix for this model is shown below and Figure 6 plots all the best performing models from each algorithm. Overall, this XGBoost model is an excellent classifier for this data. Although XGBoost did not have the best balanced accuracy metric of the four models, it had the best balance between correctly classifying true EADs as EADs and true non-EADs as non-EADs and all metrics were above 0.96. It had almost perfect specificity, only misclassifying 14 non-EADs and sensitivity of 0.964 – only misclassifying 21 EADS. Therefore, the XGBoost model was deemed the best performing model. Once the XGBoost model was recognized as the best performing model, parameter importance was explored on this model with the aid of SHAP plots.



CONFUSION MATRIX

Predicted \ Actual	0	1
0	562	14
1	21	52465

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.964	1	0.976	0.964	0.97

Accuracy	Kappa
0.999	0.969

Figure 6: Confusion matrix for XGBoost model without weights or data sampling. XGBoost was the best performing model. EADs are the positive class (Class 0) and non-EADs are the negative class (Class 1)

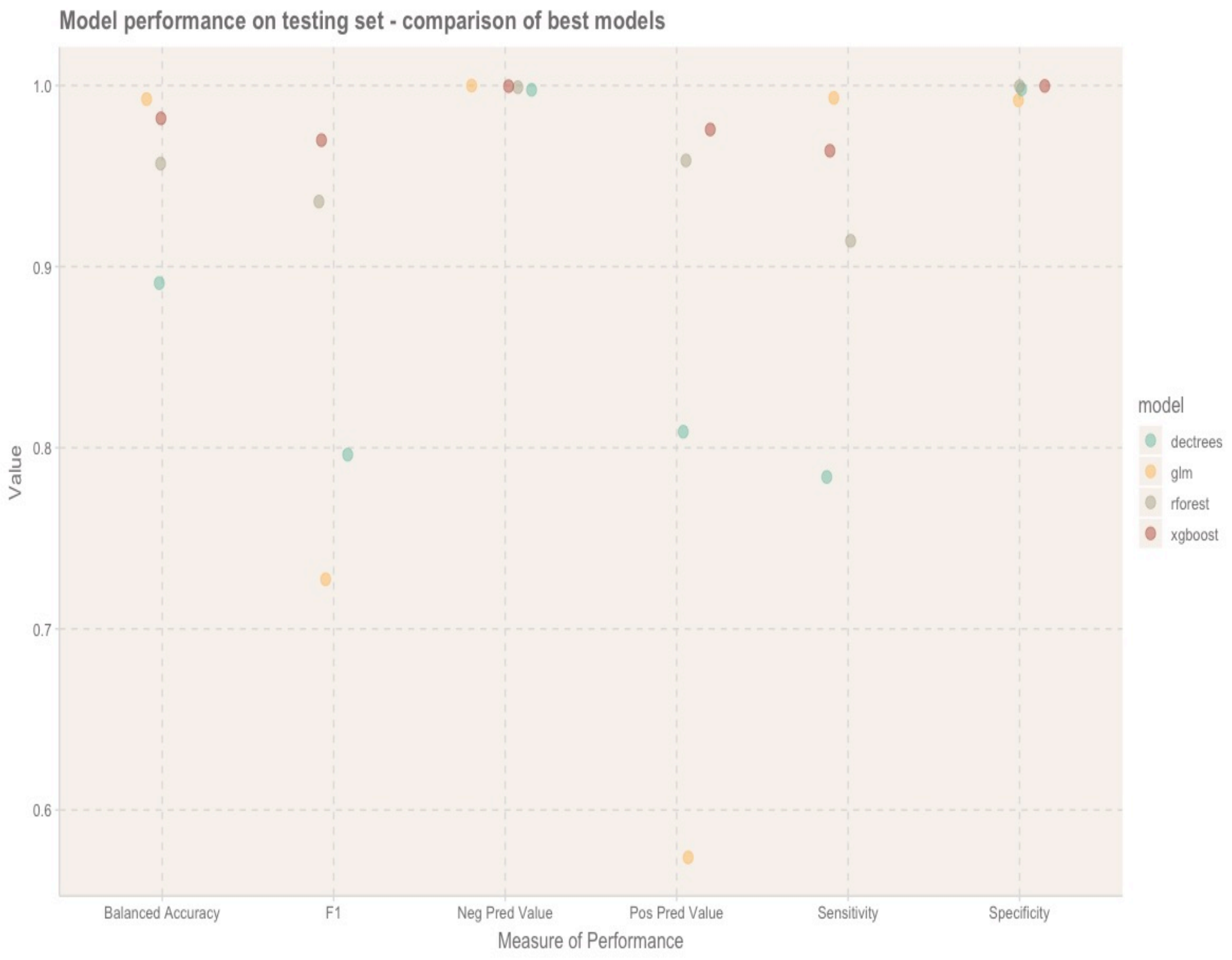


Figure 7: Plot compares metrics (from confusion matrices) for the top 4 models

SHAP summary plot

Figure 8 shows the range and distribution of the global impact that each parameter has on the model's prediction and how each parameter's value relates to its impact [31]. SHAP values are on the x axis and parameters are on the y axis – ranked by their global importance (most important first). Each dot is an observation coloured by the value of the parameter and vertical stacking of these dots result in areas with higher density when there are many dots corresponding to the same SHAP value for that specific parameter. As SHAP values increase, the log odds of predicting an EAD decrease. As SHAP values decrease, the log odds of predicting an EAD increase. Additionally, observations that move further away from the baseline of zero have stronger influence on the prediction of either EADs or non-EADs.

Unsurprisingly, the summary plot shows that the most influential parameters are G_{Kr} and $P_{Ca.}$ They display opposite behaviours, i.e., high values of P_{Ca} are associated with EADs and low values of G_{Kr} are mostly associated with EADs. This supports the well-researched phenomena of EAD genesis being supported by a block/decrease in I_{Kr} which allows for premature permeability of $I_{Ca,L}$ which can depolarize the cell and trigger an EAD [5][34][35][36].

Both high and low values of G_{Kr} (mostly high) were associated with a prediction of non-EADs. Low values of G_{Ks}, G_{K1} and G_{to} were also associated with EADs. Additionally, high values of P_{Ca}, G_{NCX} and G_{pCa} were associated with EADs. G_{Na} was quite unique. G_{Na} values near a scale factor of 1 were associated with non-EADs and the largest values of G_{Na} tended towards EADs before smaller values became more associated (and more important to) with EADs. G_{to} and G_{pCa} were also not seen as being very important to model predictions although G_{to} behaved similar to other potassium conductance parameters.

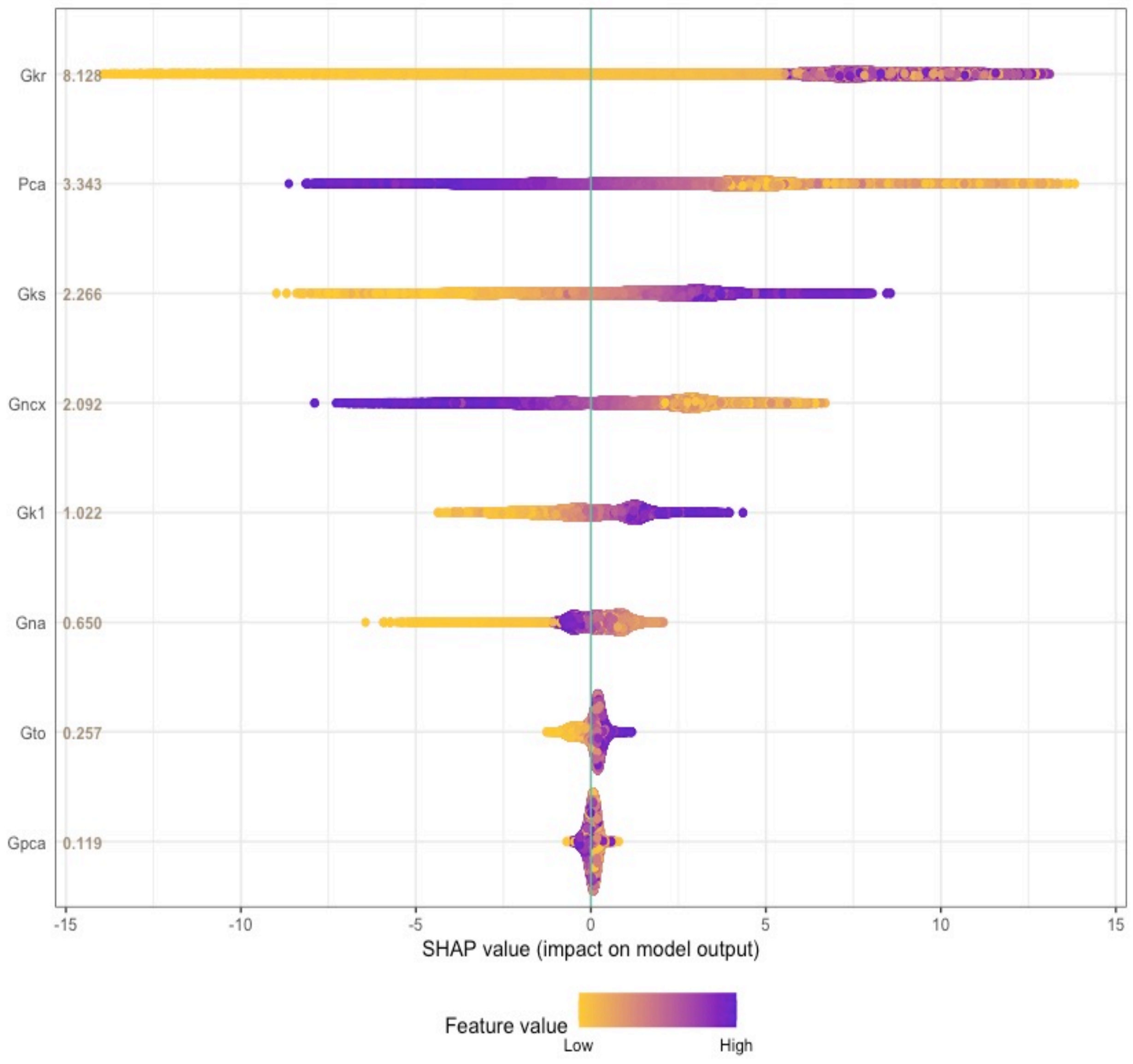


Figure 8: SHAP summary plot ranking parameter importance in the XGBoost model. Increase in SHAP values denote an increase in the log odds of predicting a non-EAD. Decrease in SHAP values denote an increase in the log odds of predicting an EAD.

SHAP dependence plots

Figure 9 shows how a specific parameter's importance changes as the value of the parameter is varied. The values of a single parameter are on the x axis and this parameter's SHAP values are on the y axis. Plots revealed that when potassium conductance decreased, the log odds of EAD genesis increased. Log odds of EAD genesis increased with ~ scale factors $G_{Kr} \leq 0.4$, $G_{Ks} \leq 1.9$ and $G_{K1} \leq 1.9$. For G_{to}, the log odds of EAD genesis increased as G_{to} decreased, however, there was a slight deviation in the trend between G_{to} between ~ 0.8 and 1 where the log odds of non-EADs increased very slightly. The unique behaviour of G_{Na} was shown in the plot below where we see a non-linear relationship. Below ~0.7 G_{Na} we see a sharp increase in log odds of EAD genesis, above this value EAD genesis decreases as G_{Na} increases but at a slower rate. The log odds of EAD genesis increased for ~ scale factors of $P_{Ca} \geq 0.5$, $G_{NCX} \geq 0.6$ and $G_{pCa} \geq 1$.

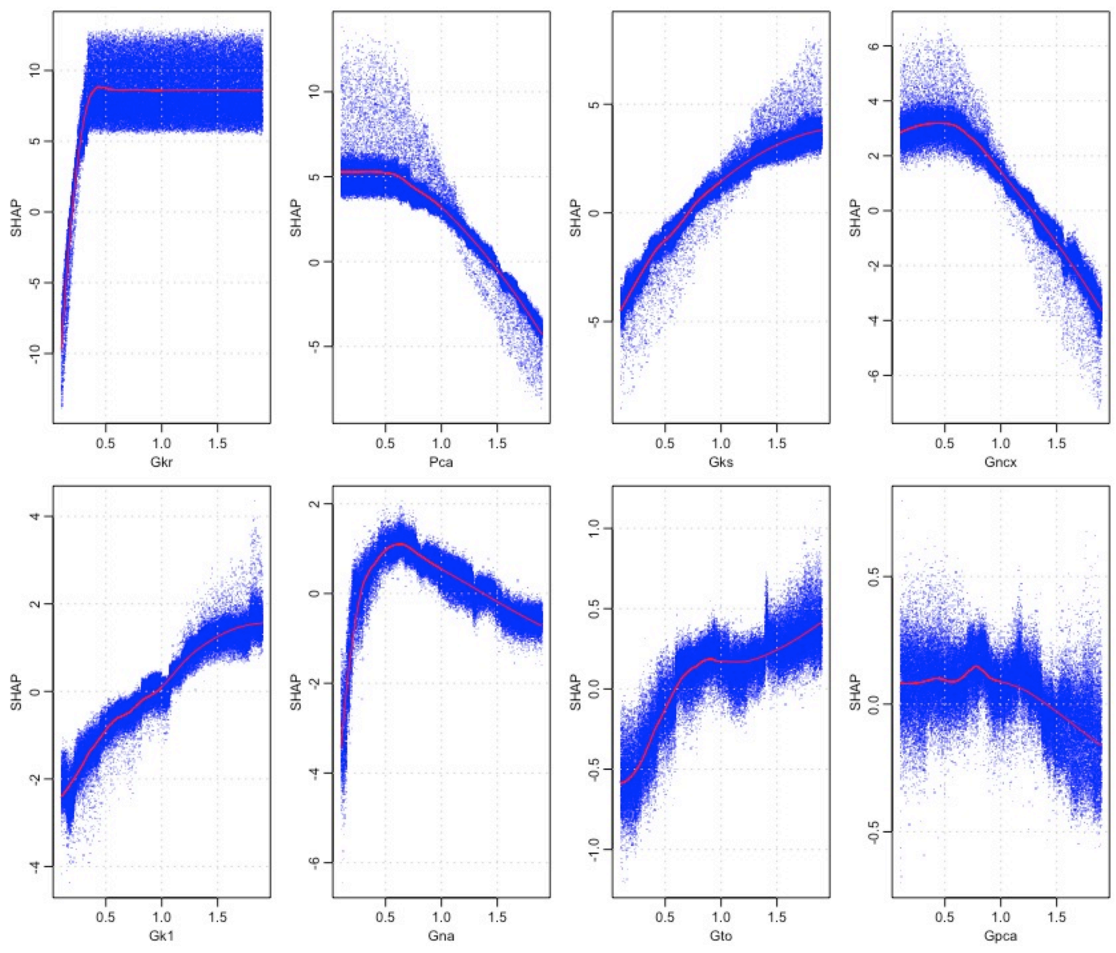


Figure 9: SHAP dependence for each of the 8 parameters

DISCUSSION

Overall, this study proved that machine learning is a great technique for exploring EAD dynamics. XGBoost is an excellent classifier for distinguishing between EAD and normal action potentials. Random Forests was a close second. XGBoost paired with SHAP plots revealed that G_{Kr} , P_{Ca} , G_{Ks} and G_{NCX} were the most important parameters in predicting EAD genesis. SHAP dependence plots further revealed that blocking potassium currents increased the probability of observing EADs while increasing conductances such as P_{Ca} and G_{NCX} increased the probability of observing EADs. Initial exploratory analysis also revealed interactions that increased EAD genesis such as low G_{Ks} and low G_{Kr}, high P_{Ca} and G_{NCX} and low G_{Kr} and G_{NCX}.

Our findings also confirm that an I_{Kr} block and sufficient inward $I_{Ca,L}$ play important roles in EAD genesis and are in fact the most influential currents in EAD genesis.

More generally, this study supports previous research that state that both a decrease in repolarizing currents and an increase in depolarizing currents promote EAD genesis. EAD genesis requires two phases involving inward and outward currents: an initiation phase and a depolarization phase.

Initiation phase - EAD genesis begins with a sufficient decrease/block of outward potassium currents. Results from this project suggest that I_{Kr} and I_{Ks} have two of the most important roles in EAD genesis and research suggest that they have major roles in repolarizing the cell [3]. G_{to} and G_{K1} also showed similar behaviour although they were not as important to the model as G_{Kr} and G_{Ks}. If G_{Kr} and/or G_{Ks} is slowed down or decreased enough, the action potential duration will lengthen during the plateau phase [5][34][35][36]. On its own, reduced potassium conductance is not enough to generate an EAD, by prolonging the action potential, it allows for two main things: 1) an increase in Ca^{2+} transients (transient increase in intracellular calcium ions) [34] 2) premature reactivation of inward channels allowing for the movement of inward currents such as $I_{Ca,L}$ [34] during phase 2 or 3 which can potentially depolarize the cell, triggering EADs.

Depolarization phase through $I_{Ca,L}$ – The L-type calcium current ($I_{Ca,L}$) is time and voltage dependent and has a ‘window’ [2] [37] (an area of overlap of the inactivation and activation relationship between ~ -30 and 0mV). If potassium conductance is too slow or decreased enough during the plateau phase, $I_{Ca,L}$ can increase enough during this window to meet the positive feedback criterion needed to depolarize the cell during repolarization and cause an EAD [3][5][35][36][38][39]. Hence, it was not a surprise that P_{Ca} was the second most important parameter in the model behind G_{Kr}.

Depolarization phase through I_{NCX} and $I_{Ca,L}$ – In a normal action potential, calcium ions will enter the cell mainly through $I_{Ca,L}$ but also through the outward I_{NCX} and T-type Ca^{2+} current (I_{CaT}) [4]. When any of them do, this influx of Ca^{2+} into the cell can potentially trigger additional release of calcium into the cell from the sarcoplasmic reticulum (the calcium store of the cell) through ryanodine receptors. If Ca^{2+} is spontaneously released from the SR during repolarization, it can increase the movement of calcium currents such as $I_{Ca,L}$ and calcium sensitive inward currents such as forward I_{NCX} (which moves 3 Na^{+} into the cell and one Ca^{2+} in its forward mode) enough to cause an EAD [3][5][34]. Since I_{NCX} is calcium sensitive, $I_{Ca,L}$ can also release excess calcium which increases forward I_{NCX} enough to cause an EAD. Additionally, increased I_{NCX} during repolarization can prolong repolarization enough to allow $I_{Ca,L}$ channels to recover and trigger another EAD [3][5].

These discoveries and studies such as the one done by Zhao et al. [40] using the patch-clamp method on rabbit ventricular myocytes have shown that these depolarization mechanisms can coexist, and they actually work in synergy [3][40]. Our findings suggest this as well as EAD presence was high during the P_{Ca} , G_{NCX} interaction when both conductances were at their highest. Since they work synergistically, it's plausible that blocking one channel is enough to prevent an EAD. In fact, researchers [34] have found that by blocking $I_{Ca,L}$ or by preventing intracellular Ca^{2+} overload (such as by blocking $I_{Ca,L}$), EADs can be suppressed. As such, since overload of the SR is thought to be one of the causes of spontaneous release of Ca^{2+} from the SR, by decreasing inward Ca^{2+}, we can prevent this overload and prevent EADs as a result.

Time was one of the biggest limitations of this study. Given more time it would be beneficial to vary more parameters in the model then utilise the XGBoost model for classification before analysis with SHAP. It's strongly recommended to use additional plots from the shap library available in Python, specifically interaction plots. As we can see, in all of the SHAP dependence plots, there are areas with a large range of SHAP values corresponding to single parameter values which suggest interactions. The shap package [33] should be used instead of the one built for R as it has more plots that can be utilised and it's also more user-friendly. Using multi-class classification may also reveal new insights. For example, performing classification where the classes are non-EADs, phase 2 EADs and phase 3 EADs. This might reveal interesting phenomena. According to a study by Enno et al. [41] in tissues with normal gap junction coupling, phase-2 EADs cannot propagate, however, phase-3 EADs can propagate under these circumstances. This is because phase-3 EADs have lower take-off potential and higher EAD amplitude. This shows that there are differences between phase-2 and phase 3 EADs and their ionic mechanisms may differ as well. A single EAD cannot cause arrhythmias, EAD propagation can lead to arrhythmias therefore, classification of EADs using a tissue model would also be beneficial.

REFERENCES

[1] Huang, X., Song, Z. and Qu, Z., 2018. Determinants of early afterdepolarization properties in ventricular myocyte models. *PLOS Computational Biology*, 14(11), p.e1006382.

[2] Kannankeril, P., Roden, D. and Darbar, D., 2010. Drug-Induced Long QT Syndrome. *Pharmacological Reviews*, 62(4), pp.760-781.

[3] Weiss, J., Garfinkel, A., Karagueuzian, H., Chen, P. and Qu, Z., 2010. Early afterdepolarizations and cardiac arrhythmias. *Heart Rhythm*, 7(12), pp.1891-1899.

[4] Volders, P., 2000. Progress in the understanding of cardiac early afterdepolarizations and torsades de pointes: time to revise current concepts. *Cardiovascular Research*, 46(3), pp.376-392.

[5] Kurata, Y., Tsumoto, K., Hayashi, K., Hisatome, I., Tanida, M., Kuda, Y. and Shibamoto, T., 2017. Dynamical mechanisms of phase-2 early afterdepolarizations in human ventricular myocytes: insights from bifurcation analyses of two mathematical models. *American Journal of Physiology-Heart and Circulatory Physiology*, 312(1), pp.H106-H127.

[6] Sobie, E., 2009. Parameter Sensitivity Analysis in Electrophysiological Models Using Multivariable Regression. *Biophysical Journal*, 96(4), pp.1264-1274.

[7] Kurata, Y., Hisatome, I., Matsuda, H. and Shibamoto, T., 2005. Dynamical Mechanisms of Pacemaker Generation in IK1-Downregulated Human Ventricular Myocytes: Insights from Bifurcation Analyses of a Mathematical Model. *Biophysical Journal*, 89(4), pp.2865-2887.

[8] Fletcher, P., Bertram, R. and Tabak, J., 2016. From global to local: exploring the relationship between parameters and behaviors in models of electrical excitability. *Journal of Computational Neuroscience*, 40(3), pp.331-345.

[9] Ferrat, L., Goodfellow, M. and Terry, J., 2018. Classifying dynamic transitions in high dimensional neural mass models: A random forest approach. *PLOS Computational Biology*, 14(3), p.e1006009.

[10] Prinz, A., Billimoria, C. and Marder, E., 2003. Alternative to Hand-Tuning Conductance-Based Models: Construction and Analysis of Databases of Model Neurons. *Journal of Neurophysiology*, 90(6), pp.3998-4015.

[11] O'Hara, T., Virág, L., Varró, A. and Rudy, Y., 2011. Simulation of the Undiseased Human Cardiac Ventricular Action Potential: Model Formulation and Experimental Validation. *PLoS Computational Biology*, 7(5), p.e1002061.

[12] Ermentrout, B., 2020. *XPPAUT*. [online] GitHub. Available at: https://github.com/Ermentrout/xppaut [Accessed 11 September 2020].

[13] Fletcher, P., 2020. *clODE*. [online] GitHub. Available at: https://github.com/patrickfletcher/clODE [Accessed 11 September 2020].

[14] Fletcher, P., 2020. *Xpptoolbox*. [online] GitHub. Available at: https://github.com/patrickfletcher/xppToolbox [Accessed 11 September 2020].

[15] H. He and E. Garcia, "Learning from Imbalanced Data", *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009. Available: 10.1109/tkde.2008.239 [Accessed 11 September 2020].

[16] B. Pes, "Handling Class Imbalance in High-Dimensional Biomedical Datasets", *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2019. Available: 10.1109/wetice.2019.00040 [Accessed 11 September 2020].

[17] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data", *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 92-122, 2012. Available: 10.1007/s10618-012-0295-5 [Accessed 11 September 2020].

[18] N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002. Available: 10.1613/jair.953 [Accessed 11 September 2020].

[19] M. Kuhn. The Caret Package. (2012). [Online]. Available: http://cranr-project.org/web/packages/caret/caret.pdf

[20] T.Hastie, J. Friedman and R. Tisbshirani, *The Elements of statistical learning*. New York: Springer, 2017.

[21] C. Metz, "Basic principles of ROC analysis", *Seminars in Nuclear Medicine*, vol. 8, no. 4, pp. 283-298, 1978. Available: 10.1016/s0001-2998(78)80014-2 [Accessed 11 September 2020].

[22] Z. MH and C. G, "Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine", *PubMed*, 2020. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8472349/. [Accessed: 11 September 2020].

[23] L. Breiman, *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001. Available: 10.1023/a:1010933404324 [Accessed 11 September 2020].

[24] K. Murphy, *Machine learning*. Cambridge, Mass.: MIT Press, 2013.

[25] T. Chen and C. Guestrin, "XGBoost", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. Available: 10.1145/2939672.2939785 [Accessed 11 September 2020].

[26] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets", *PLOS ONE*, vol. 10, no. 3, p. e0118432, 2015. Available: 10.1371/journal.pone.0118432.

[27] A. Ben-David, "About the relationship between ROC curves and Cohen's kappa", *Engineering Applications of Artificial Intelligence*, vol. 21, no. 6, pp. 874-882, 2008. Available: 10.1016/j.engappai.2007.09.009 [Accessed 11 September 2020].

[28] A. Roth, "Introduction to the Shapley value", *The Shapley Value*, pp. 1-28, 1988. Available: 10.1017/cbo9780511528446.002 [Accessed 11 September 2020].

[29] S. Lundberg and S. Lee, "An unexpected unity among methods for interpreting model predictions", *arXiv.org*, 2016. [Online]. Available: https://arxiv.org/abs/1611.07478. [Accessed: 11- Sep- 2020].

[30] [12]S. Lundberg et al., "Explainable AI for Trees: From Local Explanations to Global Understanding", *arXiv.org*, 2020. [Online]. Available: https://arxiv.org/abs/1905.04610. [Accessed: 11- Sep- 2020].

[31] S. Lundberg, G. Erion and S. Lee, "Consistent Individualized Feature Attribution for Tree Ensembles", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1802.03888. [Accessed: 11- Sep- 2020].

[32] Y. Liu, *SHAPforxgboost*, GitHub, 2020. [Online]. Available: https://github.com/liuyanguu/SHAPforxgboost. [Accessed: 11 September 2020].

[33] S. Lundberg, *shap*, GitHub, 2020. [Online]. Available: https://github.com/slundberg/shap. [Accessed: 11 September 2020].

[34] P. Milberg et al., "Blockade of ICa suppresses early afterdepolarizations and reduces transmural dispersion of repolarization in a whole heart model of chronic heart failure", *British Journal of Pharmacology*, vol. 166, no. 2, pp. 557-568, 2012. Available: 10.1111/j.1476-5381.2011.01721.x [Accessed 11 September 2020].

[35] C. January, J. Riddle and J. Salata, "A model for early afterdepolarizations: induction with the Ca2+ channel agonist Bay K 8644.", *Circulation Research*, vol. 62, no. 3, pp. 563-571, 1988. Available: 10.1161/01.res.62.3.563 [Accessed 11 September 2020].

[36] C. Luo and Y. Rudy, "A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction.", *Circulation Research*, vol. 68, no. 6, pp. 1501-1526, 1991. Available: 10.1161/01.res.68.6.1501 [Accessed 11 September 2020].

[37] C. January, V. Chau and J. Makielski, "Triggered activity in the heart: cellular mechanisms of early after-depolarizations", *European Heart Journal*, vol. 12, no., pp. 4-9, 1991. Available: 10.1093/eurheartj/12.suppl_f.4 [Accessed 11 September 2020].

[38] C. January and J. Riddle, "Early afterdepolarizations: mechanism of induction and block. A role for L-type Ca2+ current.", *Circulation Research*, vol. 64, no. 5, pp. 977-990, 1989. Available: 10.1161/01.res.64.5.977 [Accessed 11 September 2020].

[39] D. Guo, X. Zhao, Y. Wu, T. Liu, P. Kowey and G. Yan, "L-Type Calcium Current Reactivation Contributes to Arrhythmogenesis Associated with Action Potential Triangulation", *Journal of Cardiovascular Electrophysiology*, vol. 18, no. 2, pp. 196-203, 2007. Available: 10.1111/j.1540-8167.2006.00698.x [Accessed 11 September 2020].

[40] Z. Zhao et al., "Revisiting the ionic mechanisms of early afterdepolarizations in cardiomyocytes: predominant by Ca waves or Ca currents?", *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 302, no. 8, pp. H1636-H1644, 2012. Available: 10.1152/ajpheart.00742.2011 [Accessed 11 September 2020].

[41] E. de Lange, Y. Xie and Z. Qu, "Synchronization of Early Afterdepolarizations and Arrhythmogenesis in Heterogeneous Cardiac Tissue Models", *Biophysical Journal*, vol. 103, no. 2, pp. 365-373, 2012. Available: 10.1016/j.bpj.2012.06.007 [Accessed 11 September 2020].