An Investigation of Random Forest Variable Selection Algorithms

Rebecca Salzer

Fall Term 2022

Introduction

Innovations in data science within the last few decades have given researchers looking to cure diseases, such as Alzheimers and different forms of cancers, a whole collection of different techniques to help understand and eventually cure these diseases. Often this research is focused around genetic data. Genetic datasets are often incredibly large, making it nearly impossible for researchers to decode this information by hand.

New non-parametric statistical algorithms have allowed these researchers to process and better understand their data without relying on any initial parameter settings, meaning that only few assumptions are needed to be made about the data's underlying structure. One of these non-parametric algorithms is Breiman's Random Forest algorithm, created in 2001. Random forests are able to process large amounts of data in a relatively short amount of time, and create importance scores for each predictor variable included in the dataset. The variables with higher importance scores are more likely to be important when building a model to accurately predict a response.

Despite these importance scores giving researchers useful information about which variables are likely to be the most important, these importance scores alone are not a reliable measure for choosing all important variables in the dataset. There is no reliable way to set a threshold across different types of data, to determine which variable importance scores are high enough to be considered important. Scholars have since created algorithms which use these importance scores and a variety of different statistical methods, to determine which variables are actually important.

Six of these algorithms, Vita (Janitza, Celik & Boulesteix 2016), Boruta (Kursa & Rudnicki 2010), Altmann (Altmann et al. 2010), VarSelRF (Díaz-Uriarte & Alvarez de Andrés 2006), VSURF (Genuer, Poggi & Tuleau-Malot 2010) and RFE (Darst, Malecki & Engelman 2018), along with the original permutation scores (Breiman 2001) will be compared in this report, using four different datasets. The first dataset is a real dataset that trys to predict hospital deaths using numerous variables describing the patient and hospitals conditions. The next dataset is a simulated dataset where none of the variables have effect sizes and therefore none are intended to be important. The third dataset is a simulated dataset where there is real correlation between predictor variables, but only 24 of the variables have an effect on the response variable and the final dataset is also a simulated dataset where 24 variables have designated effects on the response, however there is no correlation.

Random Forests

Random forests are a machine learning algorithm, created by Breimann in 2001, that requires the user to make few assumptions about the distribution of the data and they work for both classification and regression problems (Breiman 2001). Random forests are created from a collection of hundreds to thousands of decision trees and are used as a way to get importance scores for each predictor variable included in the dataset (Schonlau & Zou 2020). When building the random forest to obtain these variable importance scores, each tree is grown using a different bootstrap sample, so some data is not used when building the tree, this data is known as out-of-bag data. A permutation process takes place as the forest is being developed and relies on this out-of-bag data (Janitza, Celik & Boulesteix 2016). After each individual tree in the forest is constructed, the values of a single variable using the out-of-bag data are randomly permuted. This is the process of randomly shuffling the variable, to break all correlation and therefore effect between this variable and the response.

After the permutation is done the out-of-bag permuted data is dropped down the tree that is being worked on and the classifications that were made using the out-of-bag permuted data are saved. This process is repeated for every variable in the dataset. Once this has been completed, the saved classifications are compared with the true class labels to get misclassification rates. The percent increase in misclassification rates after the permutation of each variable is recorded for each tree (Breiman 2001). This process is repeated thousands of times, for each of the thousands of trees as the forest is grown. The average percent increase across the entire forest in misclassification rates after the permutation of each variable in each tree is then calculated. Finally, the importance scores that are shown and the corresponding rankings are the result of the average of those increases in misclassification rates after permutation (Breiman 2001). This means that the larger the permutation importance, the more predictive the variable. This is because a large importance score indicates that when a given variable's effect on the response is removed through the permutation process, there was a large decrease in the accuracy of the model, indicating that it was a very predictive variable (Chen & Ishwaran 2012).

It should be noted that this process should be used with some caution as it contains a biased split selection (Janitza, Celik & Boulesteix 2016). This means that variables with multiple categories or splits are more likely to be deemed relevant than variables with less categories, unrelated to their true effect on the response. Using continuous data or data with similar category sizes can help avoid this bias.

Although this random forest algorithm results in permutation importance scores, it is nearly impossible to know from these importance scores alone whether a variable is relevant. Variable importance scores are made of a many different factors including things such as correlation and the number of variables being used, and some non-important variables can have positive importance scores. To solve this issue of selecting actual important variables, while using these importance scores, many algorithms have been created that use these original permutation importance scores and perform statistical procedures to determine which variables are relevant. Six of these algorithms will be examined for their accuracy and computation time in this report.

Important Variable Selection Algorithms

The algorithms being studied in this report, are algorithms that function as wrappers around random forest algorithms. They each start with the information collected from the original random forest importance scores and use statistical methods to determine whether each individual predictor is actually relevant when predicting the response. Six variable selection algorithms were examined for this report, Vita, Altmann, VarSelRF, Boruta, VSURF, RFE and then the original permutation scores. VSURF for interpretation and prediction was researched and briefly examined, but was not part of the results section, due to it's very large computation time.

Original Permutation Scores

The original permutation scores method is known to be the most naive way of selecting which variables are important (Janitza, Celik & Boulesteix 2016). To select the important variables, variable importance scores from the original random forest are obtained and variables with permutation scores above a certain threshold are considered important. Determining a threshold is arbitrary, as variable importance scores are made of a collection of different factors including things such as correlation and the number of variables being used. This means that there is no universally applicable threshold. Using this arbitrary threshold therefore makes this original permutation method known for being a naive approach (Janitza, Celik & Boulesteix 2016). It was included in this report for the purpose of comparison. The threshold being used for all datasets in this report will be 0.01.

Boruta

Boruta is an algorithm named after a god of the forest in Slavic mythology that is designed to identify important variables using random forest importance values (Degenhardt, Seifert & Szymczak 2017). Boruta, was given this name because it's tactic of increasing randomness and using that randomness to find important variables, originates from the spirit of the random forest itself. The first step for the Boruta algorithm is to create new "shadow variables." These variables are unimportant variables that will later be used for deciding which variables are really important. The shadow variables are created by adding copies of all the original variables in the dataset (Kursa & Rudnicki 2010). After the copies are made, the values of these new shadow variables are then randomly permuted, to remove all correlation with the response. This means that any impact these shadow variables are shown to have on the response will be due only to noise. Studying the results from these randomized variables allows for a better understanding of all the random impacts and correlations in the dataset, making it easier to see which of the original variables are truly unimportant, when they are showing similar random fluctuations to the shadow variables (Kursa & Rudnicki 2010).

A random forest is built using this new dataset, including the real variables and the shadow variables, and the importance scores are collected. Then the maximum importance score from all the shadow variables is found. This maximum importance score is then used as a threshold that the original non-shadow variables are compared against (Kursa & Rudnicki 2010). The null hypothesis is that the variable being examined has the same importance score as the maximal importance of the random attributes. If this is not true, and the importance score of the real variable is significantly higher or lower than the maximal score of the random attributes, this variable is deemed either important or unimportant. If a variable has a significantly higher importance value than the maximal importance of the random attributes, it is deemed important. If a variable has a significantly lower importance value than the maximal importance of the random attributes, it is deemed unimportant (Kursa & Rudnicki 2010). Variables that don't fall into either of these categories are considered undetermined.

To test and see if a variable has a significantly higher or lower variable importance score than the maximal importance score of the random attributes, a two-sided test of equality is run. The two sided test of equality is run for each variable to see if the maximum score of the shadow attributes is equal to the importance score of each variable (Kursa & Rudnicki 2010). Significance is determined as a result of under .01 or over .99 on the two sided test. Variables that have an importance significantly higher than the maximum score of the shadow attributes are deemed important, variables that have an importance significantly lower than the maximum score of the shadow attributes are deemed unimportant and those that do not fall on the extremes of the distribution are undetermined. Variables that have been deemed as either important or unimportant are classified as such and then removed from the dataset along with their corresponding shadow attributes, so that only undetermined variables remain. This entire process of creating shadow variables, collecting importance scores, and running the two-sided test of equality to determine the real variables importance is repeated until all variables have been deemed as either important or unimportant, or until a designated number of runs have been completed (Kursa & Rudnicki 2010).

VarSelRF

VarSelRF is a simple backward elimination procedure that determines the importance of variables from the prediction accuracy of different random forests. Backward elimination is the process of starting with all variables in a model and then repeatedly removing the least important variables and fitting new models, until the final model only contains the most important variable. Because of VarSelRF's backward elimination procedure, the computation time is fairly low compared to other algorithms being examined (Speiser, Miller, Tooze & Ip 2019). To begin this process, a random forest with all variables in the dataset is built. After the first forest is built a chosen fraction of the variables is dropped from the dataset, the variables which are chosen are the ones with the lowest importance scores, and a new random forest is built. The authors of this algorithm had chosen their fraction dropped to be 0.2, claiming that it allowed for an "aggressive variable selection", however these authors are using this algorithm for genetic data, which can have hundreds if not thousands of predictors (Díaz-Uriarte & Alvarez de Andrés 2006).

This process of dropping variables by a consistent percentage and creating random forests is continued until the final random forest is built with only one variable. It is important to note that the importance scores are not recalculated after each forest is built, when deciding which variables to drop. Instead, the importance scores from the original random forest are used and a fraction get dropped at each iteration, with the lowest scores being dropped first and the highest importance score being the final variable. After all random forests have been built, their OOB error rates are examined (Díaz-Uriarte & Alvarez de Andrés 2006).

There are two different ways to choose the important variables. The first option is to simply choose the random forest with the lowest OOB error rate and deem the variables within that forest important. The second option is to choose the random forest with the least amount of variables, that has an OOB error rate within one standard error of the minimum error rate of all the forests. This will generally result in choosing a smaller number of important variables. This method of selecting a model that does not have the minimum error rate is used to increase stability and predictability. When working with data with hundreds of predictors, such as genetic datasets, the chances are good that some variables that are deemed as important will have hidden correlations between other important variables, creating redundancy and instability in the model. Therefore, selecting the model with fewer predictors can lead to more stable results across multiple trials, while still achieving an error rate that is within sampling error of the minimum OOB error rate (Díaz-Uriarte & Alvarez de Andrés 2006). However, selecting using the second option could also lead to some important variables being rejected. The first option of choosing the important variables is what will be used for this report. This is due to the datasets being used having a relatively small number of predictors and because we would like this algorithm to select all potentially important variables.

Recursive Feature Elimination

Recursive Feature Elimination or RFE functions almost exactly the same as VarSelRF. The only distinction between the two algorithms, is that after each time a fraction of the variables is dropped from the dataset, the variable importance scores are recomputed and the variables are reordered. Scholars such as Darst, Malecki and Engelmanwho, claim this method performs better than VarSelRF in complex data setting with high amounts of correlation (Darst, Malecki & Engelman 2018). They claim VarSelRF is likely to do worse in these situations, because correlation can impact the original importance scores, making non important variables that are correlated with relevant ones have higher importance scores than variables with their own effect size. If these importance scores are never recomputed as variables are dropped, VarSelRF would have to choose a model with many variables that don't have their own effect size, in order to include all the variables that do, or VarSelRF would have to choose a smaller model that doesn't contain all variables that had their own effect size, to ensure non-important variables are not chosen (Darst, Malecki & Engelman 2018).

Altmann

Altmann's variable importance algorithm is a permutation-based testing approach that works using random forests. This algorithm outputs p-values for each variable, which are then used to determine if a variable is important. To obtain these p-values, a random forest is first built using all the variables and the variable importance scores are obtained. The next step is to use permutation to get importance scores for variables in settings where the variables have no effect. To do this, Altmann randomly permutes the response vector to remove any effect on the response from the variables. Once the response vector has been randomly permuted, a new random forest is built and the importance scores for each non-important variable are obtained (Janitza, Celik & Boulesteix 2016).

These importance scores from the known non-important variables are then used to form a previously unknown null distribution centered around zero. This process of randomly permuting the response vector, building a random forest and obtaining importance scores is repeated 100 times. This means that when this process is done, each variable has 100 importance scores from non-important versions of that variable, which make up that variable's previously unknown null distribution (Janitza, Celik & Boulesteix 2016). There are two different ways that you can then use these importance scores to find p-values, the parametric approach, and the non-parametric approach. The non-parametric approach is when you obtain the p-value from the fraction of the 100 importance scores in the null distribution that are larger than the original importance score. If there is a very small number of importance scores from the null distribution that are larger than the meaningful importance score, the p-value will be small, and the variable will be deemed important. This is referred to as the non-parametric approach because it doesn't assume anything about the distribution (Janitza, Celik & Boulesteix 2016). For the parametric approach, you can assume any given distribution for the importance scores of the non-important variables, the most used are Gaussian, Log-normal and Gamma-distribution. The importance scores of the non-important variables are then used to obtain the correct parameters for the assumed distribution. Finally, the p-values are calculated as the probability of obtaining an importance score that is higher than the original meaningful importance score, given this assumed distribution (Janitza, Celik & Boulesteix 2016). The non-parametric method will be used for this report.

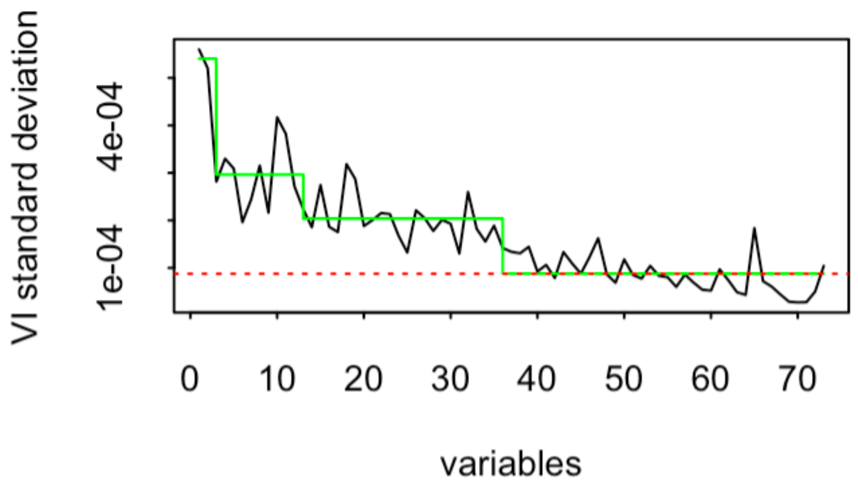
VSURF

VSURF is known as a two-step algorithm. The first step is ranking the variables based on their importance values and eliminating the unimportant ones. To rank the variables, multiple random forests are built, using all variables and the average variable importance score for each variable are collected. The variables are then ranked from highest average importance score to lowest. The first elimination process is done by keeping the variables in the order of their current rankings and plotting them on a graph (Genuer, Poggi & Tuleau-Malot 2010).

The standard deviation and the variable importance rank are plotted for each variable, with the standard deviation being on the y axis. The standard deviation of the variable importance measures should be larger when variables are actually important, therefore the values in the graph should form a curve. This graph is used to find a threshold for which variables are important, however there are many other ways to choose a threshold (Genuer, Poggi & Tuleau-Malot 2015). The most used method, however, is to find the minimum prediction value that is given by a CART model fitting this curve. The CART model fits a piece-wise constant function. A piece-wise constant function is a function defined by several constant sub-functions. The minimum prediction value given by the CART model is therefore the value of the minimum constant sub-function. This minimum prediction value is then used as a threshold for determining which variables are unimportant (Genuer, Poggi & Tuleau-Malot 2010). Figure one shows a graph displaying the curve formed by the variable importance standard deviations and rankings, along with the green CART function and the red dotted line, which represents the threshold point where there was the lowest constant sub-function. Once the threshold value is found, variables with an average variable importance value that is lower than that threshold are eliminated (Genuer, Poggi & Tuleau-Malot 2015).

Figure One

Figure one is displaying the original ranking of the variables' permutation importance scores, against their variable importance standard deviation.



The second step of this two-step algorithm is variable selection. There are two separate ways to go about the variable selection process, variable selection for interpretation or variable selection for prediction. When doing variable selection for interpretation a collection of RF models are created. The first model created is the model that contains only the most important variables, the second model contains the first and second most important variables and so on, until the final model contains all the variables retained from step one (Genuer, Poggi & Tuleau-Malot 2015). The OOB error rates for each model are computed and this process is repeated about 25 to 50 times, so that the average OOB error rates for each model can be calculated. Finally, the model with the lowest OOB error rate after this process is chosen as the best model for interpretation, and the variables within it are deemed relevant (Genuer, Poggi & Tuleau-Malot 2015). Although VSURF for prediction takes a long time as well, this is the most time consuming part of the algorithm.

The second method for the variable selection process is variable selection for prediction. In this process, variables are only added to the model if they improve the OOB error, significantly more than you would by adding unimportant noisy variables. Once again, the process starts with the ordered sequence of variables retained in step one, then one at a time variables are added to the model (Genuer, Poggi & Tuleau-Malot 2010). The OOB error of the version of the model with and without each variable are compared and the variable is only included if the decrease in the OOB error is greater than a threshold.

The threshold is there to make sure that the difference in OOB error is not due simply to noise (Genuer, Poggi & Tuleau-Malot 2015). This threshold is calculated by first collecting the OOB errors from the model that was selected for interpretation and the model with all variables that were found after the initial elimination process. The difference between the OOB errors of these two models is found and the absolute values of those differences is calculated. Finally, the threshold is set as the average of these values. After all random forests have been built and all variables have been considered for inclusion, using the threshold and the decrease in OOB error rates, the best model is the final model, with all the variables that passed the threshold. These variables are therefore deemed as important for prediction (Bag, Gupta & Deb 2022).

Vita

The Vita algorithm is designed to perform best on data with large amounts of predictor variables and a large number of non-important variables. Specifically, it was designed to perform well with genetic data. This method is not based on the classical permutation variable importance, but instead based on the hold-out version of the classical permutation importance. This allows Vita algorithm to run on datasets with thousands of predictors in a relatively small amount of time. The first step for this approach is to randomly split the dataset in half, with each half being used to create a separate random forest. The dataset is split in half so that each half can be used for creating a forest and each half can be used for calculating the importance scores of the other forest. This is known as two-fold cross validation (Janitza, Celik & Boulesteix 2016).

The two separate forests are created, and variable importance values and rankings are computed. Then, a null distribution is created using the variable importance values that are either zero or negative. This null distribution is created by first, finding the distribution of variable importance values that are either zero or negative. The next step is to mirror this distribution around the y-axis, the original distribution combined with the mirrored distribution should then create a null distribution that is centered around zero, as long as there is a large enough number of non-important variables (Janitza, Celik & Boulesteix 2016). The original Vita report shows, after running multiple simulations, that this null distribution can be assumed to follow a normal distribution. However, it also showed this process only works when using two-fold cross validation (Janitza, Celik & Boulesteix 2016). When using out-of-bag validation, the null distribution has been shown to be positively skewed and therefore not symmetric around zero (Degenhardt, Seifert & Szymczak 2017).

Once the null distributions is created, using the zero and negative importance scores from the two random forests, the positive variable importance values are compared to this distribution, to determine which variables are important. This process is done by calculating p-values, depending on where the positive variable importance scores fall on the null distribution. As variable importance scores get further from zero, they are further out in the distribution, meaning they have more extreme importance values, they will have smaller p-values and they will be more likely to be important variables. This means that variables with higher importance rankings will always have smaller p-values than variables with lower importance rankings (Janitza, Celik & Boulesteix 2016).

Like the authors of Vita mentioned, this algorithm does not perform well on data with a small amount of zero or non-important variables. If there are not enough of these variables, then the null distribution being built with them will be skewed, resulting in bad selections for important variables. There can also be highly correlated settings, where Vita won't work at all due to a complete lack of zero or negative importance scores. For this report I've created a algorithm that contains Vita, which allows Vita to function even in these circumstances.

New Vita

To ensure that Vita always functions, it is important to ensure that there are enough zero and negative importance scores to build the null distribution. The larger the sample size of these zero and negative importance scores, the more sensitive and therefore the better the null distribution of the non-important variables becomes. Therefore, this new Vita algorithm first performs a check to see how many of the variables are negative or non zero. If there are more than 30 of these variables, Vita runs as normal. If these are not more than 30, a certain number of shadow variables are created, using the same method as Boruta. The number 30 was chosen as an arbitrary threshold when determining how many variables should be sufficient. More research should be done into whether there is a better threshold that can be applied to all datasets, or a better way to determine a different thresholds based on features of the dataset being used. The amount of shadow variables that need to be made is (30-(the current number of zero and negative variables))*2.25. The difference is being multiplied by 2.25 because the shadow variables are expected to vary around zero, meaning only about half of them will result in negative or zero importance scores. The difference was multiplied by 2.25 instead of 2, to ensure that there would be a minimum of 30 variables in almost all cases.

After these shadow variables have been created, by adding copies of old variables and permuting the vectors, the permutation scores of the dataset are re-evaluated. Once these new permutation scores are obtained, shadow variables that had positive permutation scores were removed from the dataset, while the negative shadow variables remained. Finally, the original Vita process of building the null distribution, was done using the negative and zero permutation importance scores from the real dataset and the shadow variables. Then, in the same manner as Vita, p-values were determined for the variables with positive importance scores, using this distribution.

Data Exploration

The data that I will be using to investigate these algorithms is a patient survival prediction dataset from Kaggle. This dataset initially contained 85 variables and 91,713 observations, with the response variable being hospital deaths. Hospital deaths is a binary variable, with 0 meaning the subject survived and 1 meaning the subject died. The original data contained some NA values. When cleaning the data, 34,778 rows were removed which contained missing values, after one column was removed due to almost all of it's values being NA. Next, the variables' variances were checked for zero and near zero variance. Zero variance is when all of a predictors unique value's are the same and near zero variance is when the majority of a predictors unique values are the same. For this report, predictors with a ratio between their most common unique value and their second most common unique value, of at least 92.5:7.5 were considered to have near zero variance. Variables with zero or near zero variance were removed from the dataset. These included stay type, arf apache, gcs unable apache, aids, cirrhosis, hepatic failure, immunosuppression, leukemia, lymphoma and solid tumor with metastasis.

After all the variables with near zero variance were removed along with the NA's, all variables except for the response, were converted to numeric. Then, all predictor variables were centered and scaled and the response variable was converted to a factor. Finally, the dataset was reduced to it's first 10,000 observations, to lower computation time. After this data cleaning, the resulting dataset has 74 variables and 10,000 observations. The predictor variables measure various features of the hospital and the subject. These variables include things such as the hospital id, the patients age, weight, max and min heartrate in the first hour and day, height and more. A full description of the variables can be found in Appendix A. The response variable for this dataset, which will be used for prediction, is hospital deaths.

Methods

In order to test these algorithms in a variety of settings, the algorithms will be tested on three simulated datasets, in addition to the real data. Using simulated data is important so it can be known which variables are truly important. Knowing the true importance of variables is not possible while using real world data sets (Janitza, Celik & Boulesteix 2016). In addition to simulating which variables are actually important, it is also critical to investigate how these algorithms perform on different levels of correlated variables. When looking at highly correlated real world data it can become more difficult for these statistical tests to decide which variables are actually having an impact on the response, rather than being highly correlated with an important variable, leading them to choose unnecessary variables. For example, if height and weight are selected as important, bmi will also likely be selected, due to the high correlation between these variables. However, we wouldn't want our model to select bmi, as it provides no additional information about the patient, since bmi can be calculated using height and weight. Some algorithms have a much harder time dealing with high levels of correlation.

A similar process was followed for creating these simulated datasets as was in the Vita report (Janitza, Celik & Boulesteix 2016). Each of the three datasets were created from the original hospital deaths dataset, using it's design matrix, but creating new response vectors and permuting variables as necessary to create the desired simulated datasets. The first dataset simulated a scenario where none of the predictors were important. The second datset simulated a scenario where 24 of the 73 predictors had positive effect sizes, and used the original correlation patterns of the real world data. The third and final dataset simulated a scenario where 24 of the 73 predictors had positive effect sizes, but all of the predictor variables in this dataset were uncorrelated (Janitza, Celik & Boulesteix 2016).

To create the first simulated dataset, where none of the predictors are important, the first step is to use the real world data and design matrix of dataset one. In order to alter this dataset to make it so that none of the predictors are important, the elements of the response vector were permuted randomly, to destroy any associations between the elements of the response vector and the predictor variables. In this simulated dataset there is no permutation being done of the predictor variables, so all original correlation patterns between predictors from the real world data set one are still present (Janitza, Celik & Boulesteix 2016).

To create the second simulated dataset where 24 of the 74 predictors had their own effect sizes, while using the original correlation patterns of the real world data, we again start by using the original hospital deaths dataset. In this scenario again, there is no permutation being done of the predictor variables, so all original correlation patterns from the real world data set are still present. However, unlike the first simulated dataset, for this simulated dataset the response vector is not being permuted randomly, instead a new response vector was created, following a specified relation.

To create this new response vector, the first step was to randomly choose 24 predictor variables from the dataset without replacement. Based on the order they were chosen, they were each given a specific effect size. The different effect sizes were {-3, -2, -1, -0.5, 0.5, 1, 2, 3} with the first three predictor variables randomly chosen being assigned an effect size of -3 and the last three predictor variables randomly being chosen being assigned an effect size of 3. All other predictor variable effect sizes were 0. Using these effect sizes as coefficients, the probability for hospital death was calculated for each observation. The equations for calculating these exact probabilities is shown below. These probabilities are then manipulated using a binomial distribution, resulting in a response vector of 0's and 1's which were generated based on the variable effect sizes chosen (Janitza, Celik & Boulesteix 2016). By simulating the data this way, you are able to easily know which variables were intended to be important.

$$\begin{gathered} p = \\ 3 * (apachepostoperative + bmi + d1sysbpmax) + \\ 2 * (d1respratemin + d1spo2min + age) + \\ 1 * (d1heartratemax + h1respratemin + ventilatedapache) + \\ 0.5 * (apache2diagnosis + h1spo2max + d1potassiummax) + \\ (-0.5) * (gcsverbalapache + preiculosdays + patientid) + \\ (-1) * (encounterid + mapapache + d1diasbpnoninvasivemax) + \\ (-2) * (h1diasbpnoninvasivemax + h1sysbpmin + h1heartratemin) + \\ (-3) * (d1mbpmax + d1glucosemax + d1spo2max) \end{gathered}$$

$$probabilities = \frac{e^p}{1 + e^p}$$

The third and final dataset simulated a scenario where 24 of the 73 predictors were important, with varying specified effect sizes, but all of the predictor variables in this dataset were uncorrelated. This dataset was once again built from the original hospital deaths dataset. Building the response vector to ensure that 24 predictors have specific effect sizes was done the exact same way for the third simulated dataset as it was for the second simulated dataset. The same 24 variables where chosen for simulated datasets two and three.

Before creating a new response vector it was also necessary to randomly permute the values of each predictor variable independently, this process breaks associations and the correlation between the predictor variables. After permuting the variables and adding in the new response vector, the result is the final dataset with uncorrelated predictors that have a known effect size on the response (Janitza, Celik & Boulesteix 2016).

The 24 variables randomly chosen to have effect sizes for the second and third simulated datasets were, apache post-operative, bmi, day 1 sysbp max, day 1 resprate min, day 1 spo2 min, age, day 1 heartrate max, hour 1 resperate min, ventilated apache, apache 2 diagnosis, h1 spo2 max, d1 potassium max, gcs verbal apache, pre icu los days, patient id, encounter id, map apahce, day 1 diasbp noninvasive max, hour 1 diasbp noninvasive max, hour 1 sysbp min, hour one heartrate min, day 1 mbp max, day 1 glucose max and day 1 spo2 max.

The real dataset and three simulated datasets will be used in this report to test the ability of these algorithms to choose the important variables in a reasonable amount of time. For the real data and simulated datasets two and three, models will be built using each algorithms recommended variables. Using these models, model accuracy (classification rate), type one error, type two error and area under the ROC curve (AUC) will be measured. Model accuracy compares the predictions from the models that were created from the variables selected by the different algorithms, to the original real responses. Model accuracy is the percent of all cases that were accurately classified, meaning real deaths were predicted as deaths and real survival was predicted as survival. Type one and two error are two different types of missclassification and these results are once again obtained by comparing the predictions of the models created to the original real responses. Type one error is predicting that someone died when they actually survived and type two error is predicting someone survived when they actually died. AUC measures the areas under the ROC curve. The ROC curve is a probability curve that measures the true positive classification rate against the model's false positive classification rate. Measuring the area under this curve tells us how well the model is actually doing at fitting the data and making predictions. These additional measure are beneficial because sometimes model accuracy can lead to misleading results, due to skewed data. However, if this is the case with the real data or our simulated datasets, we will be able to more accurately see how the algorithms performed by comparing model AUC and type one and two error.

These results will not be measured for the first simulated dataset, because in that simulation the response vector is completely random and we do not expect any of the models to perform well. In addition to these measures of accuracy, all algorithms will be tested for computation time using all four of the datasets. Finally, the report will investigate which variables were chosen in the simulated settings where the true importance is known.

For the first simulated dataset, we know that no predictors are important, therefore the algorithms should be choosing zero variables when using the simulated dataset one. When looking at the simulated dataset three, there are exactly 24 variables with nonzero effect sizes. Since there is no correlation between predictors in this model, those 24 variables should be the only variables chosen as important. Understanding which variables we expect to be chosen for the simulated dataset two is more difficult due to the correlation between the predictors. There is currently disagreement in the field of data science, surrounding whether choosing variables as important, which have no effect size of their own, but are highly correlated with a variable with an effect size, is a good or bad thing. Some authors define relevant predictors as not only the variables that directly impact the response, but also the ones that are associated with the response due to correlation with a predictor that has it's own effect (Gregorutti, Michel & Saint-Pierre 2017). However, other scholars have specifically designed their algorithms to only select predictors that have their own effect and some others don't even define what it means for a variable to be relevant. There is currently no clear consensus in the field on what the true effect of correlation on these importance measures means (Darst, Malecki & Engelman 2018).

This lack of consensus likely stems from scholars having different reasons for using variable selection. If someone is simply looking to improve the overall accuracy of a model, they would want to include all predictors that increase that accuracy, even if they don't have their own effect. However, if someone is trying to understand which variables have the most direct impact on the response, the selection of highly correlated variables that do not have their own effect would be undesirable (Darst, Malecki & Engelman 2018). For example, in medical research if someone is trying to determine which risk factors are directly linked to a disease, including factors that have no impact on the disease but are correlated to the risk factors could be unhelpful.

For the second simulated dataset, the important predictors will be considered in this report to be those that were originally given an effect size, along with any variables that are highly correlated with an initially important variable. Variables that had a correlation of 0.6 or higher with an initially relevant variable were selected. This resulted in 49 variables being deemed as important for simulation two. This decision was made because including the predictors that are highly correlated but don't have their own effect, improves the accuracy of the models, meaning that models with these correlated predictors can more accurately predict hospital deaths. Since this is the intended goal, the report will look to see which algorithms choose all 49 important variables for this third simulated dataset.

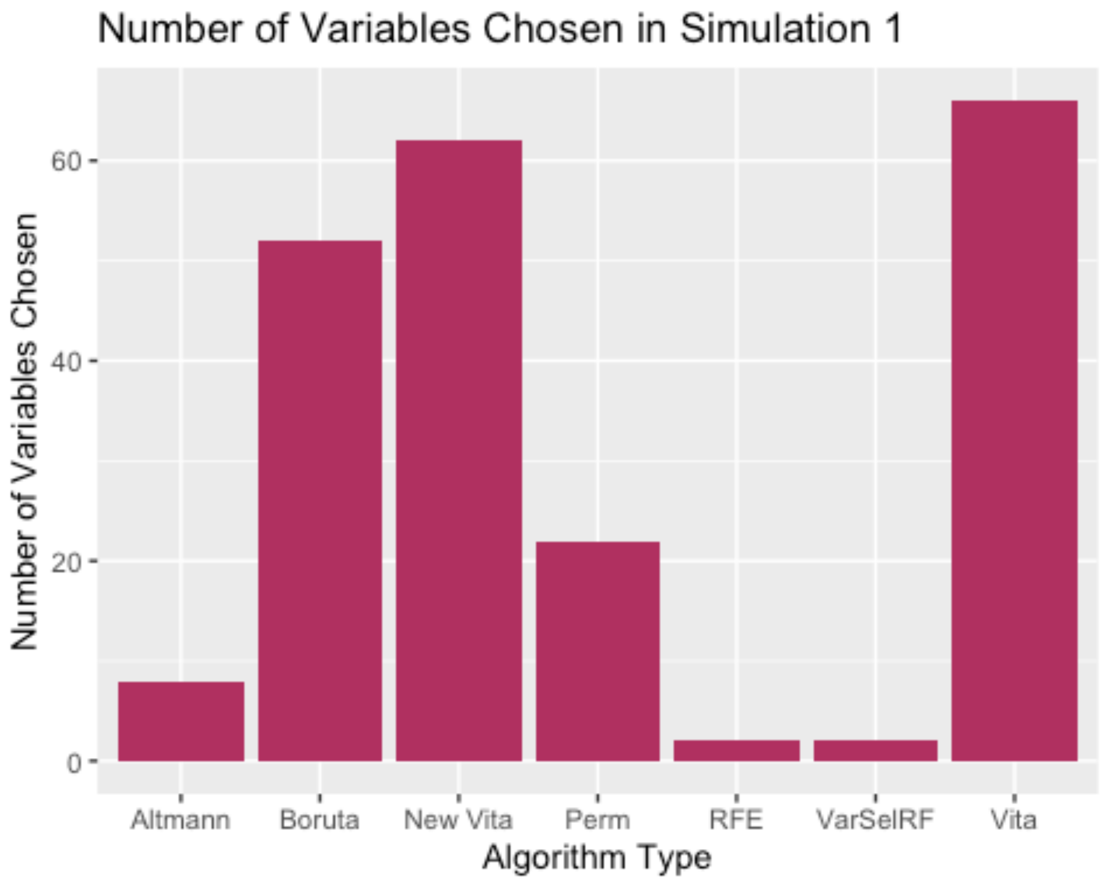
Model Accuracy, AUC, computation time, and correct variable selection will be investigated for the algorithms across the different simulations, in an attempt to see which algorithms consistently perform the best across a wide variety of datatypes. Default parameters were used for all algorithms except for VarSelRF and RFE, which had a ntree of 1000 instead of 5000, and an ntreeIterat of 500 instead of 2000. Changing these parameters made them more similar to the other models parameters, and there was not a large fear of losing too much accuracy by lowering them, because the authors of VarSelRF and RFE use a large range of parameters as examples, including ones lower than those being used in this report.

Results

VSURF was not included in the results of this report due to the algorithms long computation time. Appendix C illustrates how much longer VSURF took to perform compared to the other algorithms.

Simulation One Results

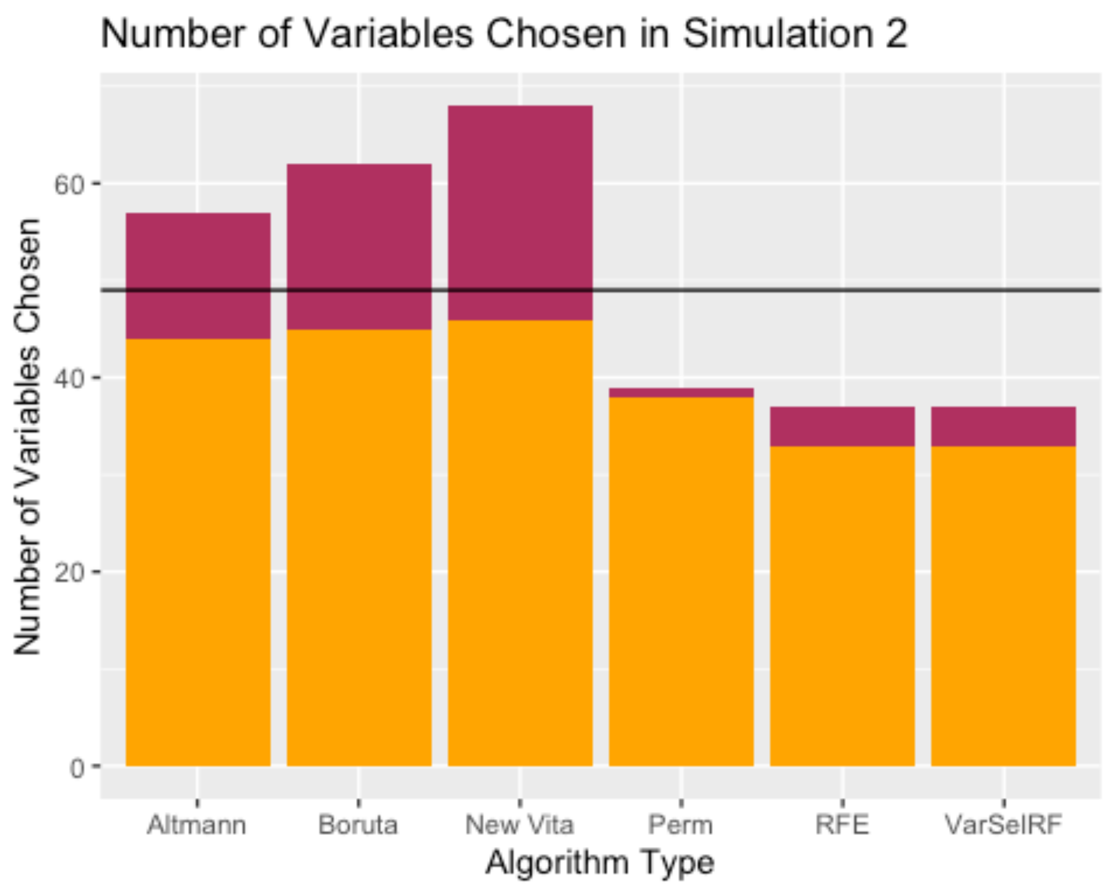
Figure Two



Simulation one was where none of the predictors were relevant, therefore algorithms who chose the smallest amount of predictors performed the best. Figure two shows that some algorithms performed much better than others on data with no relevant predictors. RFE and VarSelRF performed the best in this simulation, with each algorithm choosing the minimum number of variables it is allowed. RFE and VarSelRF are forced to choose at least two. Altmann and permutation both choose variables incorrectly, but not nearly as many as Boruta and the two Vita Algorithms. These three algorithms appear to have a very difficult time working with data with non relevant predictors.

Simulation Two Results

Figure Three

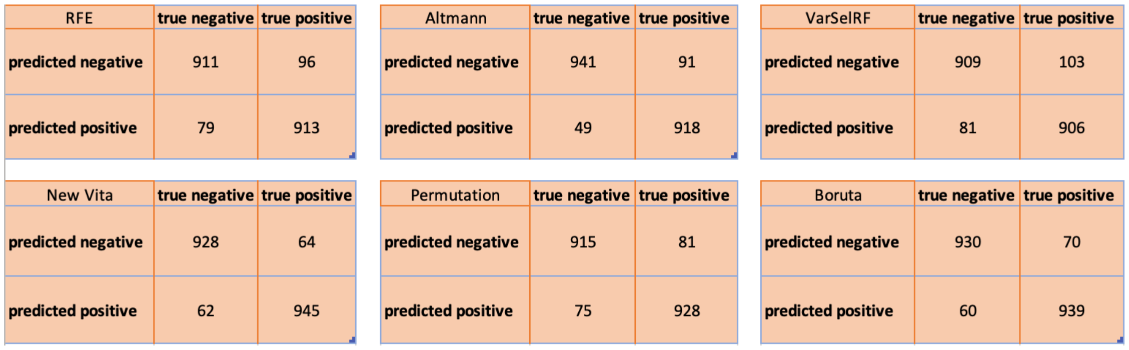


An important initial thing to notice from Figure three, is that the original Vita is not included with the rest of the algorithms. This is because, due to the correlated simulated data, when Vita calculated the initial permutation importance, none of the variables had negative or zero importance scores. This meant that Vita was unable to form a null distribution using those scores and continue the process of it's algorithm. In Figure three, the orange section of the bars represents the amount of variables that were chosen correctly, which were part of the 49 intended important variables. The maroon section of the bars represents all variables that were incorrectly chosen. The black line in the graph is at 49, to show how many variables should have been chosen for the ideal model.

It appears that Altmann, Vita and Boruta performed the best in choosing the most intended important variables, however all three choose a large amount of irrelevant variables, which can be damaging for interpretation. The new Vita algorithm got closest to choosing all the correct variables, however it also chose a the largest number of unimportant variables. Perm, RFE and VarSelRF all chose under 40 variables, however they performed the best at choosing the least amount of unimportant variables. A full list of variables that the algorithms chose as important for the second simulated dataset can be found in appendix B.

Figure four below displays the classification tables for the models created by the algorithms in simulation two. Classification tables compare the predicted results from the models, with the actual results from the simulated data. These tables help display overall accuracy as well as type one and two error. Type one error would be predicting that someone died when they actually survived and type two error would be predicting that someone survived when they actually died.

Figure Four



RFE	true negative	true positive
predicted negative	911	96
predicted positive	79	913

Altmann	true negative	true positive
predicted negative	941	91
predicted positive	49	918

VarSelRF	true negative	true positive
predicted negative	909	103
predicted positive	81	906

New Vita	true negative	true positive
predicted negative	928	64
predicted positive	62	945

Permutation	true negative	true positive
predicted negative	915	81
predicted positive	75	928

Boruta	true negative	true positive
predicted negative	930	70
predicted positive	60	939

Figure four shows that the variables chosen by the new Vita for the second simulated dataset, resulted in the most accurate model. However, this is likely because Vita choose many variables and could likely be overfitting the data. Altmann and Boruta performed similarly well, with both algorithms sharing the lowest false negative missclassification rate. RFE and VarSelRf seem to have performed the worst with this simulated dataset. Type One error, Type Two error, Accuracy and AUC values will be further examined when comparing the algorithms across all simulations.

Simulation Three Results

Figure Five

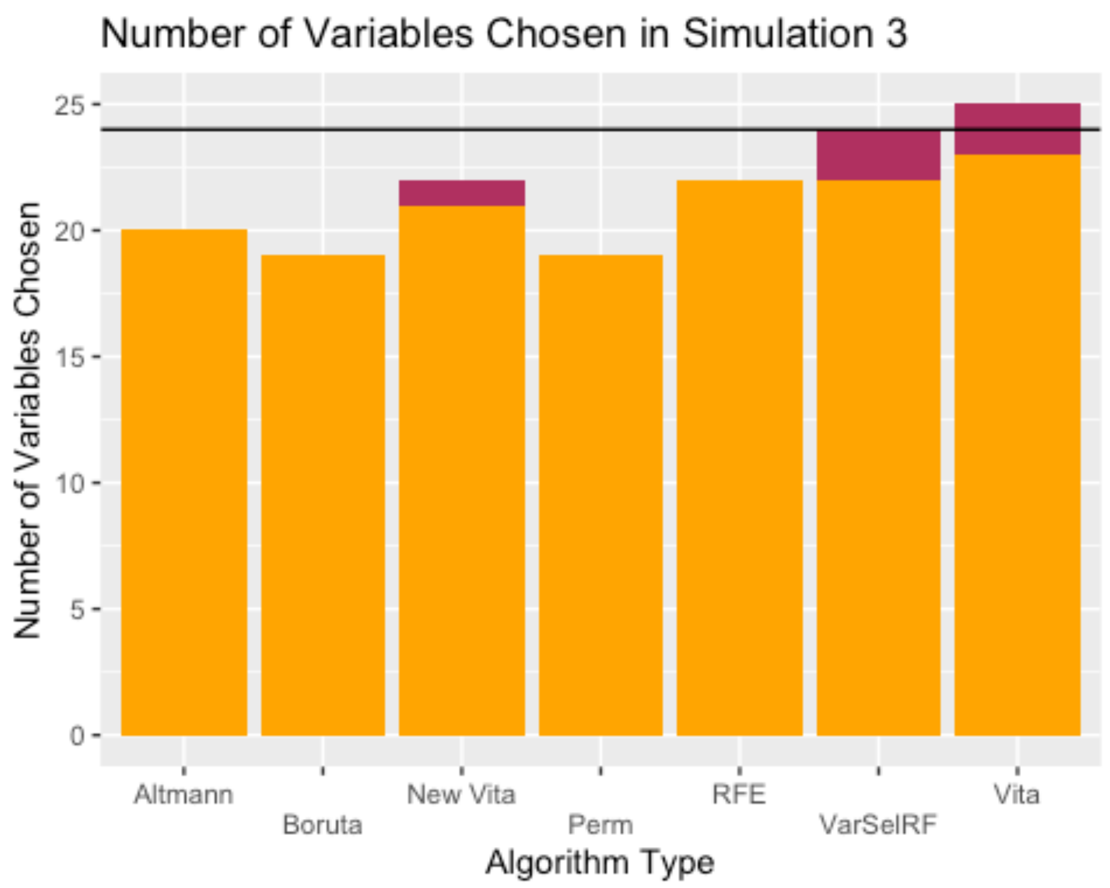


Figure five shows the variables chosen by the algorithms using the third simulated dataset, where there is no correlation. The range for how many variables each model chose is much smaller for this simulated dataset than for simulated dataset two. Each algorithm chose between 19 and 25 variables and they chose at most two incorrect variables. Altmann, Boruta, Permutation and RFE all resulted in no variables incorrectly being chosen, however they chose less correct variables than other models. All models are performing very well on this simulated dataset without correlation, and Vita appears to be doing the best at choosing the largest amount of important variables. None of the algorithms chose all 24 important variables. When looking at which of the variables these algorithms missed, the variables that were not chosen were consistently those with the smallest designated effect sizes of 0.5 or -0.5. A full list of variables that the algorithms chose as important for the third simulated dataset can be found in appendix B.

The classification tables in Figure Six show the the algorithms are doing similarly well on classification accuracy. It appears that Vita, Altmann and VarSelRF are performing the best, while Boruta and Permutation are performing the worst.

Figure Six

Vita	true negative	true positive
predicted negative	925	73
predicted positive	73	929

Altmann	true negative	true positive
predicted negative	919	73
predicted positive	79	929

VarSelRF	true negative	true positive
predicted negative	922	71
predicted positive	76	931

New Vita	true negative	true positive
predicted negative	923	81
predicted positive	75	921

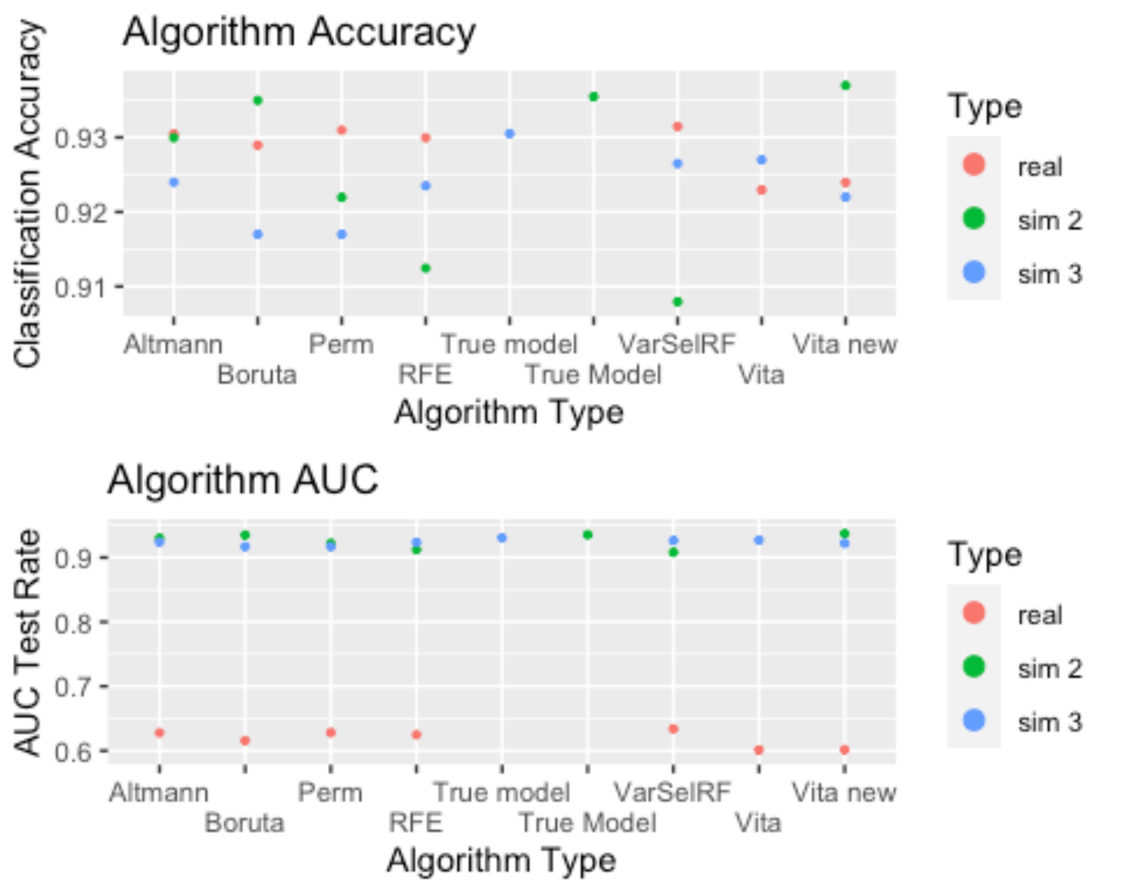
RFE	true negative	true positive
predicted negative	921	76
predicted positive	77	926

Permutation	true negative	true positive
predicted negative	910	78
predicted positive	88	924

Boruta	true negative	true positive
predicted negative	910	78
predicted positive	88	924

Results Comparing Simulation Types

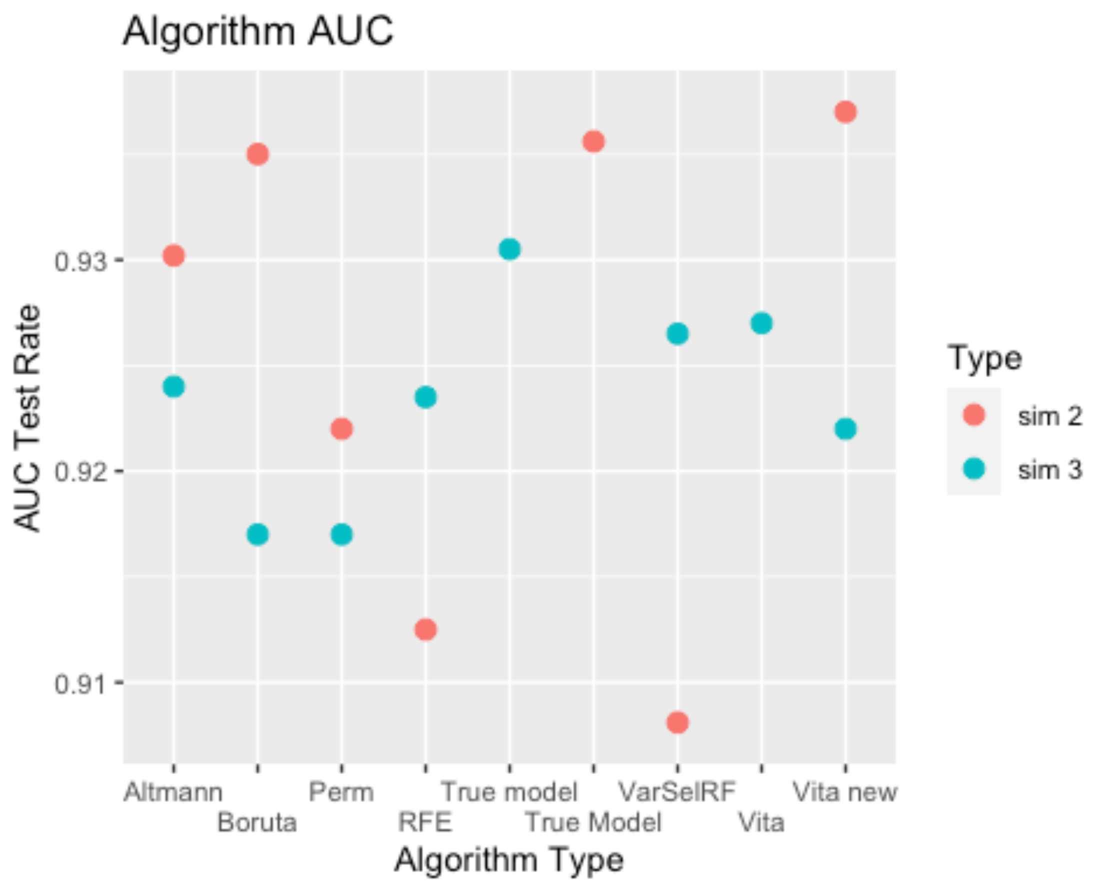
Figure Seven



From Figure seven it is clear to see that some algorithms are performing better on some datatypes than others. Some algorithms such as Boruta and Altmann are performing much better on simulation two data than simulation three, whereas models such as VarselRF and RFE have the opposite results. It appears that Altmann and the new Vita Algorithm have the highest model accuracies across all three datasets. Some algorithms selected variables are even outperforming the accuracy of the true model for simulation two, which includes all variables with non zero effect sizes and variables highly correlated with those variables. For example, for simulation two, the new Vita, which chose many more than the 49 intended important variables, has a higher accuracy than the model with only the intended variables. This was surprising and implies that some variables that have even less than a .6 correlation with the variables that have their own effect size, are providing additional information to the model and should be included if the goal is for more accurate prediction.

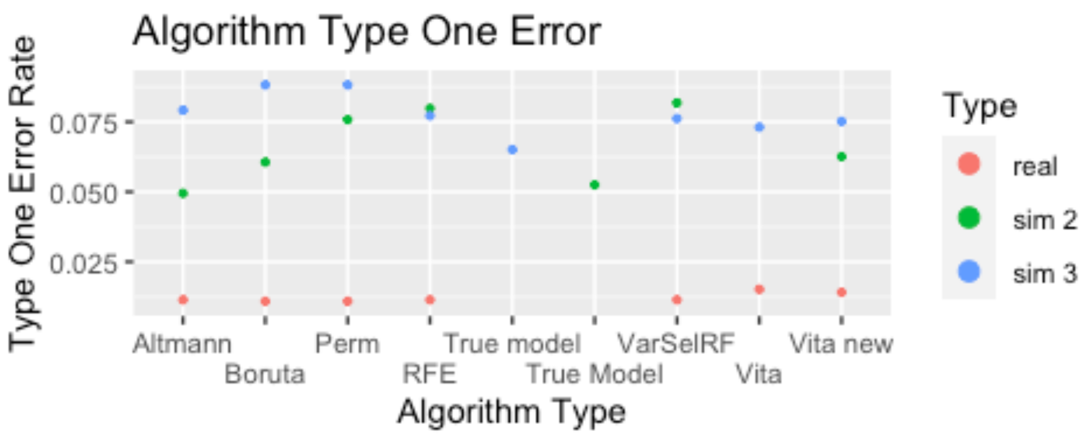
When looking at the AUC results, it is clear that all algorithms are doing much better on the two simulated datasets compared to the real dataset. The reason this was not clear when showing the accuracy, is because all algorithms chose variables for a model that could correctly predict the majority of cases where patients don't die, meaning these models have a low type one error. For the real data, the cases where patients don't die happen to be an overwhelming majority of the cases (1838 people survived and 151 died). This results in a good overall accuracy, despite the fact that all models built from the algorithms have very high type two errors. The models built can predict when a patient dies less than 30% of the time. The overall accuracy appears artificially high due to this unbalanced response vector, so the type one error is balancing out the type two error, but the AUC clearly shows that the algorithms are not doing nearly as well at choosing variables which are capable of accurately predicting hospital deaths when working with the real data. VarSelRF has the highest AUC when looking at the model built using important variables found from the real data.

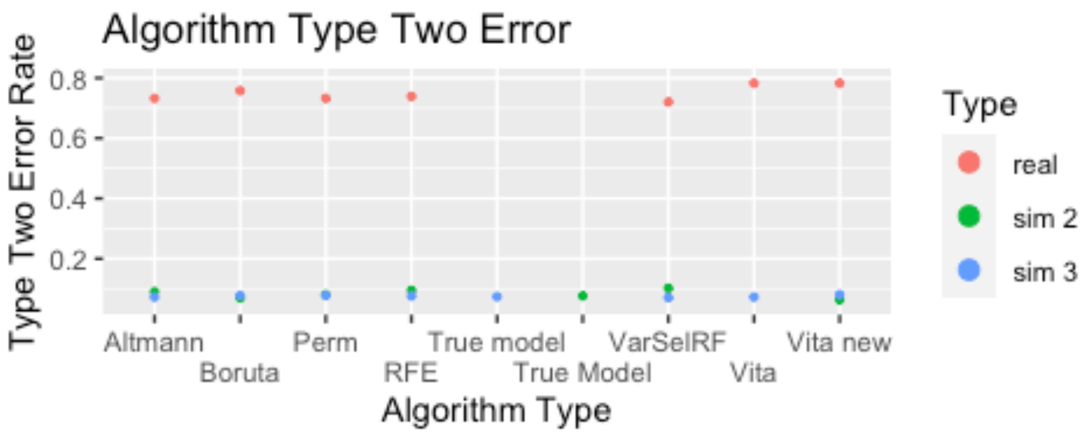
Figure Eight



For Figure eight, the real data AUC Test rates were removed, to closer examine the differences between simulation two and three. It appears that algorithms performed in a more similar manner for simulation three without correlation compared to simulation two. Vita, VarSelRF and Altmann had the highest AUC for models using the third simulated dataset. Boruta and the new Vita still had a high AUC, but comparatively did not perform as well as the other algorithms for simulation three. However, Boruta and the new Vita both performed the best using the second simulation datasets with correlation. Altmann also did very well on the second simulated dataset. It appears that overall Altmann and new Vita algorithms choose variables for models with the highest average AUC's across simulation one and two. A closer look into the type one and two errors of these models is shown below.

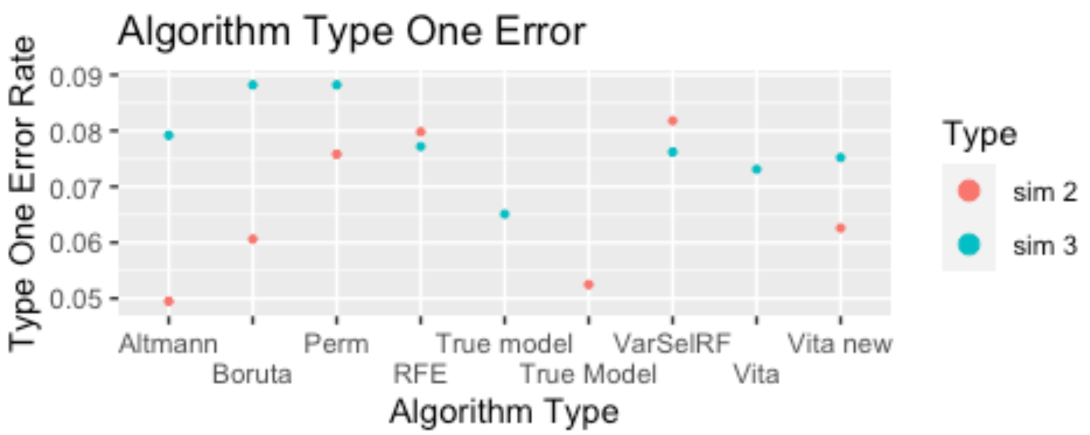
Figure Nine





When looking at Figure nine, you can once again clearly see that the real data is an outlier. The real data results in the lowest type one error of all the data types and the highest type two error, this is true across all algorithms. This is once again because the type one error is artificially low, due to having a very large number of patients who survive. Also as mentioned before, all algorithms are having a much harder time at choosing variables which can accurately predict death for the real data, and this is resulting in a very high type two error. Despite none of the algorithms doing very well, Altmann selected a model that has the lowest type two error and Altmann and the new Vita algorithm selected a model have the lowest type one errors, when working with the real data.

Figure Ten



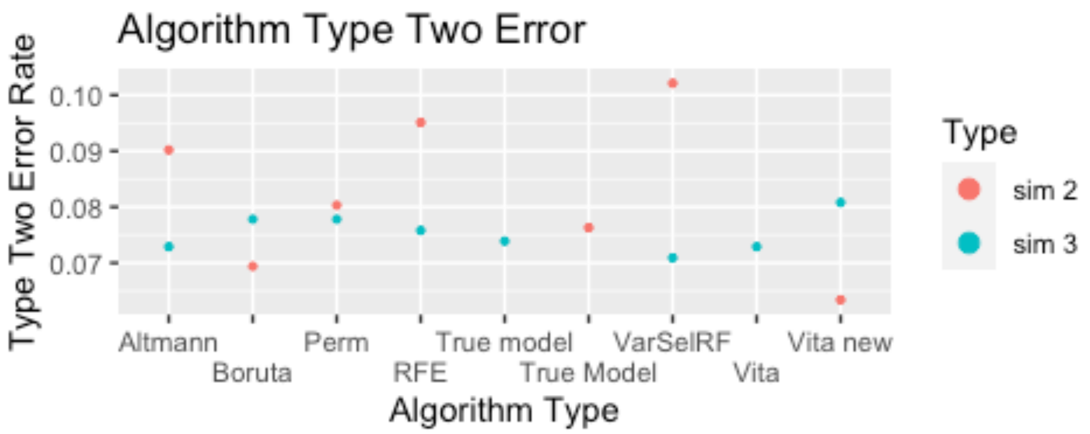
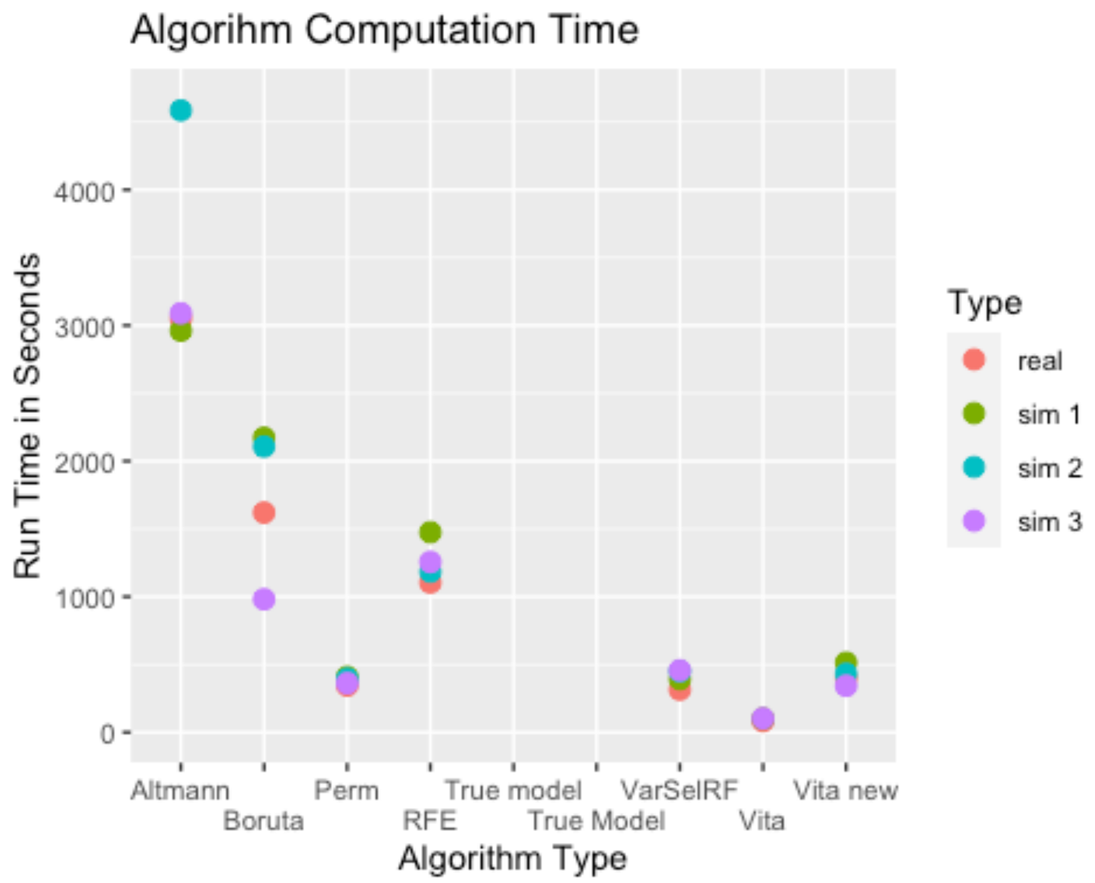


Figure ten takes a closer extermination at how the algorithms performed on the simulated datasets, with type one and type two error. For type one error, where one is incorrectly predicting a patient dies, Boruta, Altmann and the new Vita performed the best for the second simulated dataset, while Vita and VarSelRF had the lowest type one error when using the third simulated dataset. The true model for simulated datastes two and three, with 24 and 49 important predictors, had a lower type one error rate than any of the other algorithms, except for Altmann, which had a lower type one error rate than the true model using the second simulated dataset. Multiple algorithms had a lower type two error than the true model for the two simulations. Boruta and the new Vita both had very low type two errors for the second simulated dataset, meaning they were very good at accurately predicting deaths that occur. VarSelRF, Altmann and Vita had the lowest type two errors for the third simulated dataset.

After looking at the overall accuracy and classification error rates of the algorithms in the different simulations, it is now also important to take into account computation time. When working with medical or genetic datasets, computation times for these algorithms can quickly become large. For example, the VSURF algorithm can take up to 12 hours to run with this relatively small dataset while using paramaters lower than the algorithm's default settings on a MacBook, this is once again why this algorithm was not included. Because of the potential to have these large computation times, it is not only important to consider how well these algorithms are doing at selecting important variables, but also how long it is taking them, as some algorithms will not be as feasible for working with large amounts of data.

Figure Eleven



From Figure eleven we can see that the Altmann algorithm has by far the highest computation time, of about 3,300 seconds on average, with Boruta having the second highest time of around 1,600 seconds on average. RFE has the third highest time, and it is quite a bit higher than VarSelRF. This is what we would expect, because RFE is a similar algorithm to VarSelRF, except RFE recomputes importance scores every time variables are dropped, which takes a significant amount of time. The new Vita algorithm takes about twice as long as Vita for a similar reason. The new Vita algorithm has to recompute importance scores twice, when the dataset doesn't initially produce 30 negative or zero importance scores. This process of computing importance scores takes up the majority of the time of the Vita algorithm, therefore the new Vita, which has to do this step twice, takes about double the time as Vita.

Figure Twelve

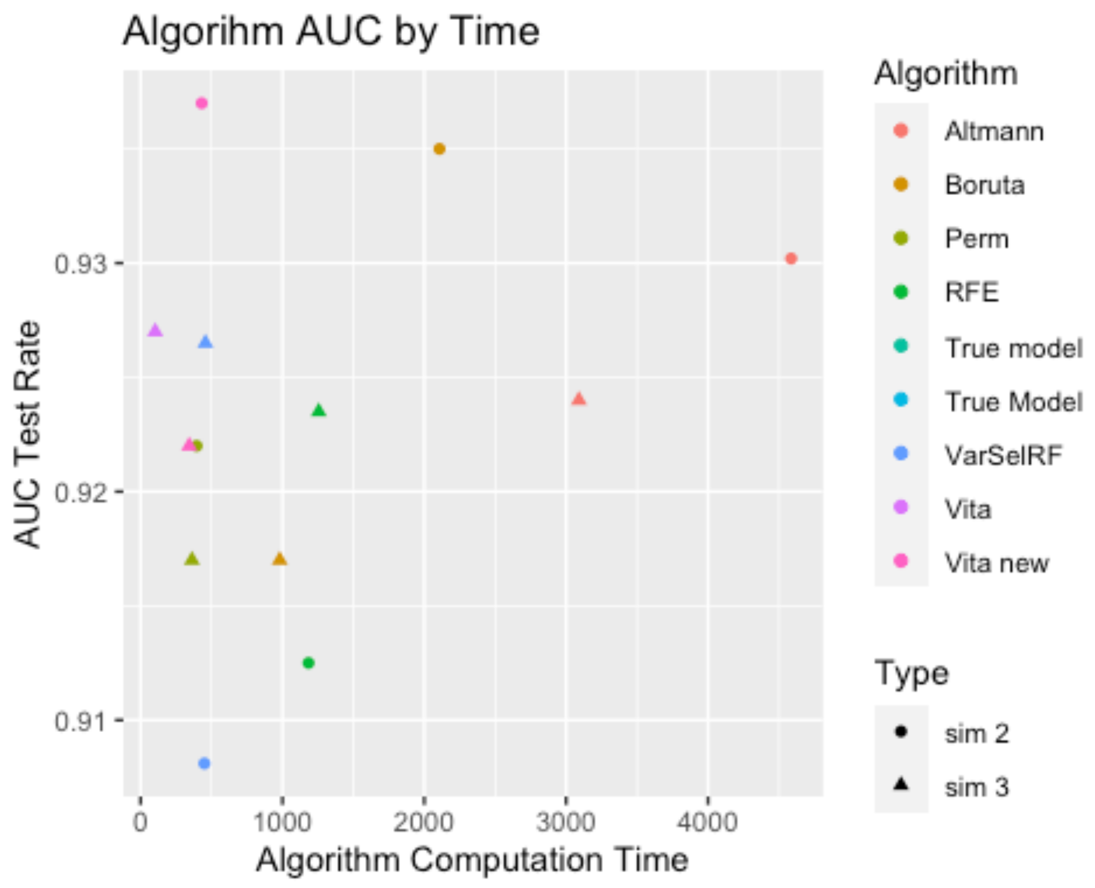


Figure twelve shows that there does appear to be a positive correlation between AUC test rate and computation time. Although the model AUC is positively correlated with algorithm computation time, it appears that Vita and the new Vita are positive outliers to this trend. Vita and the new Vita both have very high AUC's for each simulation, when taking into consideration their very low computation times. From examining these results it appears that out of the algorithms studied, Vita and the new Vita would perform the best at accurately selecting important predictors, in a short amount of time. Having a low computation time can make them very useful when working with data that has large numbers of predictors and observations, such as genetic data. However, it is also important to remember that Vita performed the worst on data without any important predictors, so if one is unsure if there are important predictors in their dataset, Vita is not the best option. Although Altmann appears to be performing very well on both simulated datasets, it's high computation time can make it difficult to use when working with large datasets.

Observations From the Real Dataset

Although it doesn't make sense to see how many variables the algorithms correctly chose for the real data, it is interesting to examine what variables were commonly chosen. Apache 4a hospital death prob and apache 4a icu death prob were chosen for every model. This makes sense as these two variables are predictors of death based on other variables.

Therefore it is a good thing that every model acknowledged that these were good predictors of death. Other variables such as d1 sysbp min, h1 mbp min, h1 sysbp max and gcs motor apache, were chosen for a majority of the algorithms. These variables that were chosen multiple times by multiple different algorithms are important variables for predicting patient death in hospital settings. An entire list of variables that were chosen by each algorithm using the real data is listed in Appendix B.

Discussion and Conclusion

After examining these results it appears that Vita is performing on average the best in simulations where there are important predictors, when taking into account computation time. Altmann is actually performing the best overall across these different simulations, but it has a much higher computation time than Vita, which can be a big determent when working with large amounts of data. Despite Vita working well on situations where there are relevant predictors, especially the second and third simulated datasets, Vita is performing very badly on the simulated dataset with no relevant predictors. In this situation, Vita and the new Vita chose nearly all predictors as relevant. This is a big downside to Vita, if you're working with data and are unsure if there are any important predictors. Altmann selected some variables as relevant for simulation one, but it was far less than Vita. Altmann also performed notably better than Vita on the real dataset, with a lower type one and two error and a higher AUC.

Depending on the size of the data one is working with and the computational capabilities available to them, it appears that it would be better to use Altmann if resources allow. However, when working with very large amounts of data, for things such as genetic research, Vita and the new Vita algorithm appear to perform the best in a small amount of time. Also, despite Altmann performing the best, it is important to once again note that none of these algorithms performed very well on the real data, this shows that there is still large room for improvement when developing these algorithms for selecting important variables. More research needs to be done on whether or not correlated variables should be selected and if they should, investigations need to take place to determine what types and levels of correlations these algorithms should determine as important.

When doing future research on which current algorithms perform best, it would be beneficial to look at a wider range of algorithms, as there has been hundreds created. It would also be beneficial to do more experiments to find the true optimal parameters for these algorithms and to see if the optimal parameters differ based on the size or the type of the data. A final area for future research would be to examine the capabilities of algorithms like VSURF, which were not able to be included in this report due to computation time. Research using a more powerful computer would be able to better examine these computationally difficult algorithms, in a similar way to what was done in this paper.

Continuing to compare these algorithms and understand their strengths and weaknesses in different settings, is a critical first step in improving these algorithms in the future. These algorithms have the potential to work through incredibly large amounts of data, unguided, and find patterns and important variables that we wouldn't have been able to see before. This is especially useful when working with medical or genetic data, and trying to find the cure for genetic diseases that we don't yet understand. As these algorithms improve, there is hope that they will be able to find the potential causes for these diseases, through variable selection and other machine learning techniques, leading to potential cures down the road.

Appendix

Appendix A: Full List of Real Dataset Variables & Descriptions

encounter id- Unique identifier associated with a patient unit stay

patient id- Unique identifier associated with a patient

hospital id- Unique identifier associated with a hospital

age- The age of the patient on unit admission

bmi- The body mass index of the person on unit admission

elective surgery- Whether the patient was admitted to the hospital for an elective surgical operation

ethnicity- The common national or cultural tradition which the person belongs to

gender- Sex of the patient

height- The height of the person on unit admission

icu admit source- The location of the patient prior to being admitted to the unit

icu id- A unique identifier for the unit to which the patient was admitted

icu type- A classification which indicates the type of care the unit is capable of providing

pre icu los days- The length of stay of the patient between hospital admission and unit admission

weight- The weight (body mass) of the person on unit admission

apache 2 dignosis- The APACHE II diagnosis for the ICU admission

apache 3j diagnosis- The APACHE III-J sub-diagnosis code which best describes the reason for the ICU admission

apache post operative- The APACHE operative status (post-operative or non-operative)

gcs eyes- The eye opening component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score

gcs motor- The motor opening component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score

gcs verbal- The verbal component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III

heart rate apache- The heart rate measured during the first 24 hours which results in the highest APACHE III score intubated apache- Whether the patient was intubated at the time of the highest scoring arterial blood gas used in the oxygenation score

map apache- The mean arterial pressure measured during the first 24 hours which results in the highest APACHE III score

resprate apache- The respiratory rate measured during the first 24 hours which results in the highest APACHE III score

temp apache- The temperature measured during the first 24 hours which results in the highest APACHE III score

ventilated apache- Whether the patient was invasively ventilated at the time of the highest scoring arterial blood gas using the oxygenation

d1 diasbp max- The patient's highest diastolic blood pressure during the first 24 hours of their unit stay

d1 diasbp min- The patient's lowest diastolic blood pressure during the first 24 hours of their unit stay

d1 diasbp noninvasive max- The patient's highest diastolic blood pressure during the first 24 hours of their unit stay, non-invasively measured

d1 diasbp noninvasive min- The patient's lowest diastolic blood pressure during the first 24 hours of their unit stay, non-invasively measured

d1 heartrate max- The patient's highest heart rate during the first 24 hours of their unit stay

d1 heartrate min- The patient's lowest heart rate during the first 24 hours of their unit stay

d1 mbp max- The patient's highest mean blood pressure during the first 24 hours of their unit stay

d1 mbp min- The patient's lowest mean blood pressure during the first 24 hours of their unit stay

d1 mbp noninvasive max- The patient's highest mean blood pressure during the first 24 hours of their unit stay, non-invasively measured

d1 mbp noninvasive min- The patient's lowest mean blood pressure during the first 24 hours of their unit stay, non-invasively measured

d1 resprate max- The patient's highest respiratory rate during the first 24 hours of their unit stay

d1 resprate min- The patient's lowest respiratory rate during the first 24 hours of their unit stay

d1 spo2 max- The patient's highest peripheral oxygen saturation during the first 24 hours of their unit stay d1 spo2 min- The patient’s lowest peripheral oxygen saturation during the first 24 hours of their unit stay

d1 sysbp max- The patient’s highest systolic blood pressure during the first 24 hours of their unit stay

d1 sysbp min- The patient’s lowest systolic blood pressure during the first 24 hours of their unit stay

d1 sysbp noninvasive max- The patient’s highest systolic blood pressure during the first 24 hours of their unit stay, non-invasively measured

d1 sysbp noninvasive min- The patient’s lowest systolic blood pressure during the first 24 hours of their unit stay, non-invasively measured

d1 temp max- The patient’s highest core temperature during the first 24 hours of their unit stay

d1 temp min- The patient’s lowest core temperature during the first 24 hours of their unit stay

h1 diasbp max- The patient’s highest diastolic blood pressure during the first hour of their unit stay

h1 diasbp min- The patient’s lowest diastolic blood pressure during the first hour of their unit stay

h1 diasbp noninvasive max- The patient’s highest diastolic blood pressure during the first hour of their unit stay, non-invasively measured

h1 diasbp noninvasive min- The patient’s lowest diastolic blood pressure during the first hour of their unit stay, non-invasively measured

h1 heartrate max- The patient’s highest heart rate during the first hour of their unit stay

h1 heartrate min- The patient’s lowest heart rate during the first hour of their unit stay

h1 mbp max- The patient’s highest mean blood pressure during the first hour of their unit stay

h1 mbp min- The patient’s lowest mean blood pressure during the first hour of their unit stay

h1 mbp noninvasive max- The patient’s highest mean blood pressure during the first hour of their unit stay, non-invasively measured

h1 mbp noninvasive min- The patient’s lowest mean blood pressure during the first hour of their unit stay, non-invasively measured

h1 resprate max- The patient’s highest respiratory rate during the first hour of their unit stay h1 resprate min- The patient’s lowest respiratory rate during the first hour of their unit stay

h1 spo2 max- The patient’s highest peripheral oxygen saturation during the first hour of their unit stay

h1 spo2 min- The patient’s lowest peripheral oxygen saturation during the first hour of their unit stay

h1 sysbp max- The patient’s highest systolic blood pressure during the first hour of their unit stay

h1 sysbp min- The patient’s lowest systolic blood pressure during the first hour of their unit stay

h1 sysbp noninvasive max- The patient’s highest systolic blood pressure during the first hour of their unit stay, non-invasively measured

h1 sysbp noninvasive min- The patient’s lowest systolic blood pressure during the first hour of their unit stay, non-invasively measured

d1 glucose max- The highest glucose concentration of the patient in their serum or plasma during the first 24 hours of their unit stay

d1 glucose min- The lowest glucose concentration of the patient in their serum or plasma during the first 24 hours of their unit stay

d1 potassium max- The highest potassium concentration for the patient in their serum or plasma during the first 24 hours of their unit stay

d1 potassium min- The lowest potassium concentration for the patient in their serum or plasma during the first 24 hours of their unit stay

apache hospital death prob- The APACHE IVa probabilistic prediction of in hospital mortality for the patient which utilizes the APACHE III score

apache icu death prob- The APACHE IVa probabilistic prediction of in ICU mortality for the patient which utilizes the APACHE III score

diabetes mellitus- Whether the patient has been diagnosed with diabetes, either juvenile or adult onset, which requires medication.

apache 3j bodysystem- Admission diagnosis group for APACHE III

apache 2 bodysystem- Admission diagnosis group for APACHE II

hosptial death- If the patient died in the hospital

Appendix B: Variables Chosen as Important by Algorithms Using Simulations Two and Three and the Real Data

Variables Chosen as Important by Algorithms Using the Real Data

VarSelRF	RFE	Permutation	New Vita	New Vita
apache 4a hospital death prob	apache 3j diagnosis	apache 4a hospital death prob	h1 diasbp nonionvasive max	d1 diasbp max
apache 4a icu death prob	apache 4a hospital death prob	apache 4a icu death prob	h1 diasbp noninvasive min	d1 diasbp min
d1 sysbp min	apache 4a icu death prob		h1 heartrate max	d1 diasbp noninvasive max
h1 mbp min	d1 diasbp min		h1 heartrate min	d1 diasbp noninvasive min
h1 sysbp max	d1 diasbp max		h1 mbp max	d1 heartrate max
h1 sysbp min	d1 diasbp noninvasive max		h1 mbp min	d1 heartrate min
h1 sysbp noninvasive max	d1 diasbp noninvasive min		h1 mbp noninvasive max	d1 mbp max
h1 sysbp noninvasive min	d1 heartrate max		h1 mbp noninvasive min	d1 mbp min
	d1 heartrate min		h1 resprate max	d1 mbp noninvasive max
	h1 mbp max		h1 resprate min	d1 mbp noninvasive min
	h1 mbp min		h1 spo2 max	d1 resprate max
	h1 mbp noninvasive max		h1 spo2 min	d1 resprate min
	h1 mbp noninvasive min		h1 sysbp max	d1 spo2 max
	d1 spo2 min		h1 sysbp min	d1 spo2 min
	d1 sysbp max		h1 sysbp noninvasive max	d1 sysbp max
	d1 sysbp min		h1 sysbp noninvasive min	d1 sysbp min
	d1 sysbp noninvasive max		ethnicity	d1 sysbp noninvasive max
	d1 sysbp noninvasive min		icu admit source	d1 sysbp noninvasive min
	d1 temp min		icu type	d1 temp max
	gcs motor apache		icu id	d1 temp min
	gsc verbal apache		pre icu los days	h1 diasbp max
	h1 diasbp nonionvasive max		weight	h1 diasbp min
	h1 diasbp noninvasive min		apache 2 diagnosis	d1 glucose max
	h1 diasbp max		apache 3j diagnosis	d1 glucose min
	h1 diasbp min		apache post operative	d1 potassium max
	h1 heartrate max		gcs eyes apache	d1 potassium min
	h1 heartrate min		gcs motor apache	apache 4a hospital death prob
	h1 mbp max		gcs verbal apache	apache 4a icu death prob
	h1 mbp min		heart rate apache	apache 3j bodysystem
	h1 mbp noninvasive max		intubated apache	apache 2 bodysystem
	h1 mbp noninvasive min		map apache	
	h1 sysbp max		resprate apache	
	h1 sysbp min		temp apache	
	h1 sysbp noninvasive max		ventilated apache	
	h1 sysbp noninvasive min		elective surgury	
	heart rate apache		hospital id	
	map apache		age	
			bmi	

Boruta	Boruta	Vita	Vita	Altmann
h1 diasbp nonionvasive max	d1 heartrate max	h1 diasbp nonionvasive max	d1 diasbp max	apache post operative
h1 diasbp noninvasive min	d1 heartrate min	h1 diasbp noninvasive min	d1 diasbp min	gcs eyes apache
h1 heartrate max	d1 mbp max	h1 heartrate max	d1 diasbp noninvasive max	gcs motor apache
h1 heartrate min	d1 mbp min	h1 heartrate min	d1 diasbp noninvasive min	gcs verbal apache
h1 mbp max	d1 mbp noninvasive max	h1 mbp max	d1 heartrate max	intubated apache
h1 mbp min	d1 mbp noninvasive min	h1 mbp min	d1 heartrate min	ventilated apache
h1 mbp noninvasive max	d1 resprate max	h1 mbp noninvasive max	d1 mbp max	apache 4a hospital death prob
h1 mbp noninvasive min	d1 resprate min	h1 mbp noninvasive min	d1 mbp min	apache 4a icu death prob
h1 resprate max	d1 spo2 max	h1 resprate max	d1 mbp noninvasive max	
h1 resprate min	d1 spo2 min	h1 resprate min	d1 mbp noninvasive min	
h1 spo2 max	d1 sysbp max	h1 spo2 max	d1 resprate max	
h1 spo2 min	d1 sysbp min	h1 spo2 min	d1 resprate min	
h1 sysbp max	d1 sysbp noninvasive max	h1 sysbp max	d1 spo2 max	
h1 sysbp min	d1 sysbp noninvasive min	h1 sysbp min	d1 spo2 min	
h1 sysbp noninvasive max	d1 temp max	h1 sysbp noninvasive max	d1 sysbp max	
h1 sysbp noninvasive min	d1 temp min	h1 sysbp noninvasive min	d1 sysbp min	
icu admit source	h1 diasbp max	ethnicity	d1 sysbp noninvasive max	
icu id	h1 diasbp min	gender	d1 sysbp noninvasive min	
pre icu los days	apache 2 bodysystem	height	d1 temp max	
weight	d1 glucose max	icu admit source	d1 temp min	
apache 2 diagnosis	d1 glucose min	icu type	h1 diasbp max	
apache 3j diagnosis	d1 potassium max	icu id	h1 diasbp min	
apache post operative	d1 potassium min	pre icu los days	d1 glucose max	
gcs eyes apache	apache 4a hospital death prob	weight	d1 glucose min	
gcs motor apache	apache 4a icu death prob	apache 2 diagnosis	d1 potassium max	
gcs verbal apache	apache 3j bodysystem	apache 3j diagnosis	d1 potassium min	
heart rate apache	age	apache post operative	apache 4a hospital death prob	
intubated apache	bmi	gcs eyes apache	apache 4a icu death prob	
map apache	elective surgury	gcs motor apache	apache 3j bodysystem	
resprate apache		gcs verbal apache	apache 2 bodysystem	
temp apache		heart rate apache	diabetes mellitus	
ventilated apache		intubated apache	hospital id	
d1 diasbp max		map apache	age	
d1 diasbp min		resprate apache	bmi	
d1 diasbp noninvasive max		temp apache	elective surgury	
d1 diasbp noninvasive min		ventilated apache		

Variables Chosen as Important by Algorithms Using the Second Simulated Dataset

Intended Model	Intended Model	Altmann	Altmann	Permutation	Permutation
encounter id	h1 heartrate max	encounter id	h1 diasbp nonionvasive max	encounter id	h1 sysbp max
patient id	h1 heartrate min	age	h1 diasbp noninvasive min	age	h1 sysbp min
age	h1 mbp max	bmi	h1 heartrate max	bmi	h1 sysbp noninvasive max
bmi	h1 mbp min	elective surgury	h1 heartrate min	elective surgury	h1 sysbp noninvasive min
elective surgery	h1 mbp noninvasive max	height	h1 mbp max	icu admit source	d1 glucose max
weight	h1mbp noninvasivemin	icu admit source	h1 mbp min	weight	
ice admit source	h1 resprate min	icu id	h1 mbp noninvasive max	apache 2 diagnosis	
pre icu los days	h1 spo2 max	weight	h1 mbp noninvasive min	apache 3j diagnosis	
apache 2 diagnosis	h1 sysbp max	apache 2 diagnosis	h1 resprate max	apache post operative	
apache 3j diagnosis	h1 sysbpmin	apache 3j diagnosis	h1 resprate min	heart rate apache	
apache post operative	h1 sysbp noninvasive max	apache post operative	h1 sysbp max	map apache	
gcs eyes apache	h1 sysbp noninvasive min	gcs eyes apache	h1 sysbp min	d1 diasbp max	
gcs motor apache	d1 glucose max	gcs motor apache	h1 sysbp noninvasive max	d1 diasbp min	
gcs verbal apache	d1 potassium max	heart rate apache	h1 sysbp noninvasive min	d1 diasbp noninvasive max	
heart rate apache	d1 potassium min	intubated apache	d1 potassium max	d1 diasbp noninvasive min	
intubated apache		map apache	d1 glucose max	d1 heartrate max	
ventilated apache		ventilated apache	apache 4a hospital death prob	d1 mbp max	
map apache		d1 diasbp max	apache 4a icu death prob	d1 mbp noninvasive max	
d1_diasbp_max		d1 diasbp min	diabetes mellitus	d1 spo2 max	
d1_diasbp_noninvasive_max		d1 diasbp noninvasive max	apache 3j bodysystem	apache 4a hospital death prob	
d1_heartrate max		d1 diasbp noninvasive min	apache 2 bodysystem	diabetes mellitus	
d1 heartrate min		d1 heartrate max	h1 diasbp max	d1 sysbp max	
d1_mbp_max		d1 heartrate min	h1 diasbp min	d1 sysbp noninvasive max	
d1_mbp_noninvasive_max		d1 mbp max		h1 diasbp max	
d1_resprate min		d1 mbp min		h1 diasbp min	
d1 spo2 max		d1 mbp noninvasive max		h1 diasbp nonionvasive max	
d1 spo2min		d1 mbp noninvasive min		h1 diasbp noninvasive min	
d1 sysbp max		d1 resprate min		h1 heartrate max	
d1 sysbp min		d1 spo2 max		h1 heartrate min	
d1 sysbp noninvasive max		d1 spo2 min		h1 mbp max	
d1 sysbp noninvasive min		d1 sysbp max		h1 mbp min	
h1_diasbp max		d1 sysbp min		h1 mbp noninvasive max	
h1 diasbp nonionvasive max		d1 sysbp noninvasive max		h1 mbp noninvasive min	
h1 diasbp noninvasive min		d1 sysbp noninvasive min		h1 resprate min	

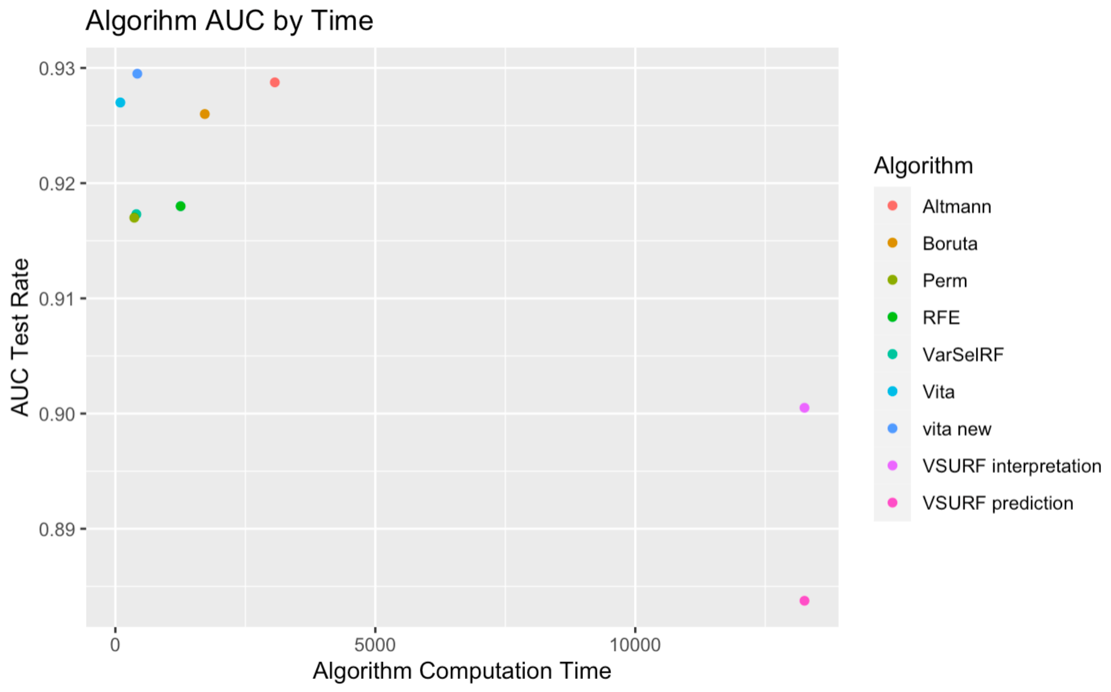
New Vita	New Vita	Boruta	Boruta	RFE	VarSelRF
h1 diasbp nonionvasive max	d1 diasbp max	h1 diasbp nonionvasive max	d1 diasbp max	age	age
h1 diasbp noninvasive min	d1 diasbp min	h1 diasbp noninvasive min	d1 diasbp min	apache 2 diagnosis	apache 2 diagnosis
h1 heartrate max	d1 diasbp noninvasive max	h1 heartrate max	d1 diasbp noninvasive max	apache 3j diagnosis	apache 3j diagnosis
h1 heartrate min	d1 diasbp noninvasive min	h1 heartrate min	d1 diasbp noninvasive min	apache 4a hospital death prob	apache 4a hospital death prob
h1 mbp max	d1 heartrate max	h1 mbp max	d1 heartrate max	apache post operative	apache post operative
h1 mbp min	d1 heartrate min	h1 mbp min	d1 heartrate min	bmi	bmi
h1 mbp noninvasive max	d1 mbp max	h1 mbp noninvasive max	d1 mbp max	d1 diasbp max	d1 diasbp max
h1 mbp noninvasive min	d1 mbp min	h1 mbp noninvasive min	d1 mbp min	d1 diasbp min	d1 diasbp min
h1 resprate max	d1 mbp noninvasive max	h1 resprate max	d1 mbp noninvasive max	d1 diasbp noninvasive max	d1 diasbp noninvasive max
h1 resprate min	d1 mbp noninvasive min	h1 resprate min	d1 mbp noninvasive min	d1 diasbp noninvasive min	d1 diasbp noninvasive min
h1 spo2 max	d1 resprate max	h1 spo2 max	d1 resprate max	d1 glucose max	d1 glucose max
h1 spo2 min	d1 resprate min	h1 spo2 min	d1 resprate min	d1 mbp max	d1 mbp max
h1 sysbp max	d1 spo2 max	h1 sysbp max	d1 spo2 max	d1 mbp noninvasive max	d1 mbp noninvasive max
h1 sysbp min	d1 spo2 min	h1 sysbp min	d1 sysbp max	d1 resprate min	d1 resprate min
h1 sysbp noninvasive max	d1 sysbp max	h1 sysbp noninvasive max	d1 sysbp min	d1 spo2 max	d1 spo2 max
h1 sysbp noninvasive min	d1 sysbp min	h1 sysbp noninvasive min	d1 sysbp noninvasive max	d1 spo2 min	d1 spo2 min
ethnicity	d1 sysbp noninvasive max	ethnicity	d1 sysbp noninvasive min	d1 sysbp max	d1 sysbp max
height	d1 sysbp noninvasive min	icu admit source	h1 diasbp max	d1 sysbp noninvasive max	d1 sysbp noninvasive max
icu admit source	d1 temp max	icu id	h1 diasbp min	elective surgery	elective surgery
icu id	d1 temp min	pre icu los days	d1 glucose max	h1 diasbp max	h1 diasbp max
icu type	h1 diasbp max	weight	d1 glucose min	h1 diasbp min	h1 diasbp min
pre icu los days	h1 diasbp min	apache 2 diagnosis	d1 potassium max	h1 diasbp nonionvasive max	h1 diasbp nonionvasive max
weight	apache 2 bodysystem	apache 3j diagnosis	d1 potassium min	h1 diasbp noninvasive min	h1 diasbp noninvasive min
apache 2 diagnosis	d1 glucose max	apache post operative	apache 4a hospital death prob	h1 heartrate max	h1 heartrate max
apache 3j diagnosis	d1 glucose min	heart rate apache	apache 4a icudeath prob	h1 heartrate min	h1 heartrate min
apache post operative	d1 potassium max	intubated apache	apache 3j bodysystem	h1 mbp max	h1 mbp max
gcs verbal apache	d1 potassium min	map apache	apache 2 bodysystem	h1 mbp min	h1 mbp min
heart rate apache	apache 4a hospital death prob	resprate apache	age	h1 mbp noninvasive max	h1 mbp noninvasive max
intubated apache	apache 4a icu death prob	ventilated apache		h1 mbp noninvasive min	h1 mbp noninvasive min
map apache	apache 3j bodysystem	encounter id		weight	weight
resprate apache	hospital id	ethnicity		h1 resprate min	h1 sysbp max
temp apache	encounter id	bmi		h1 sysbp max	h1 sysbp min
ventilated apache	age	elective surgery		h1 sysbp min	h1 sysbp noninvasive max
elective surgury	bmi	height		h1 sysbp noninvasive max	h1 sysbp noninvasive min
				h1 sysbp noninvasive min	icu admit source
				icu admit source	map apache
				map apache	heart rate apache

Variables Chosen as Important by Algorithms Using the Third Simulated Dataset

Intended Model	Vita	New Vita	Boruta
encounter id	encounter id	encounter id	encounter id
patient id	patient id	patient id	patient id
age	age	age	age
bmi	bmi	bmi	bmi
apache post operative	apache post operative	apache 2 diagnosis	apache post operative
map apache	map apache	apache post operative	map apache
ventilated apache	ventilated apache	map apache	ventilated apache
d1 diasbp noninvasive max	d1 diasbp noninvasive max	ventilated apache	d1 diasbp noninvasive max
d1 heartrate max	d1 heartrate max	d1 diasbp noninvasive max	d1 heartrate max
d1 mbp max	d1 mbp max	d1 heartrate max	d1 mbp max
d1 resprate min	d1 resprate min	d1 mbp max	d1 resprate min
d1 spo2 max	d1 spo2 max	d1 resprate min	d1 spo2 max
d1 spo2 min	d1 spo2 min	d1 spo2 max	d1 spo2 min
d1 potassium max	d1 potassium max	d1 spo2 min	d1 sysbp max
d1 sysbp max	d1 sysbp max	d1 sysbp max	h1 diasbp noninvasive max
h1 diasbp noninvasive max	h1 diasbp noninvasive max	h1 diasbp noninvasive max	h1 heartrate min
h1 heartrate min	h1 heartrate min	h1 diasbp noninvasive min	h1 sysbp min
h1 sysbp min	h1 sysbp min	h1 heartrate min	h1 resprate min
apache 2 diagnosis	h1 spo2 max	h1 sysbp min	d1 glucose max
h1 spo2 max	h1 resprate min	h1 resprate min	
h1 resprate min	d1 glucose max	d1 glucose max	
d1 glucose max	gcs verbal apache	gcs verbal apache	
gcs verbal apache	pre icu los days		
pre icu los days	heart rate apache		
	resprate apache		

RFE	Altmann	Permutation	VarSelRF
age	encounter id	encounter id	encounter id
bmi	patient id	patient id	patient id
apache post operative	age	age	age
map apache	bmi	bmi	bmi
ventilated apache	apache post operative	apache post operative	apache post operative
d1 diasbp noninvasive max	map apache	map apache	map apache
d1 heartrate max	ventilated apache	ventilated apache	ventilated apache
d1 mbp max	d1 diasbp noninvasive max	d1 diasbp noninvasive max	d1 diasbp noninvasive max
d1 resprate min	d1 heartrate max	d1 heartrate max	d1 heartrate max
d1 spo2 max	d1 mbp max	d1 mbp max	d1 mbp max
d1 spo2 min	d1 resprate min	d1 resprate min	d1 resprate min
d1 potassium max	d1 spo2 max	d1 spo2 max	d1 spo2 max
d1 sysbp max	d1 spo2 min	d1 spo2 min	d1 spo2 min
h1 diasbp noninvasive max	d1 sysbp max	d1 sysbp max	d1 potassium max
h1 heartrate min	h1 diasbp noninvasive max	h1 diasbp noninvasive max	d1 sysbp max
h1 sysbp min	h1 heartrate min	h1 heartrate min	h1 diasbp noninvasive max
patient id	h1 sysbp min	h1 sysbp min	h1 heartrate min
h1 resprate min	h1 resprate min	h1 resprate min	h1 sysbp min
d1 glucose max	d1 glucose max	d1 glucose max	h1 resprate min
gcs verbal apache	gcs verbal apache		d1 glucose max
pre icu los days			d1 sysbp min
encounter id			h1 spo2 max
			h1 sysbp noninvasive min
			pre icu los days

Appendix C: Algorithm AUC by Time with VSURF included



It is important to note here that this graphic is meant to illustrate VSURF's bad computation time, it is not intended to illustrate how VSURF is performing. Due to computation time, suboptimal parameters were chosen when running VSURF's algorithm. It is expected when using the optimal parameters, that VSURF would have performed better with this data, however it would also further increase VSURFS computation time.

Sources

Agarwal, Mitisha. "Patient Survival Prediction." Kaggle, 26 Dec. 2021, https://www.kaggle.com/datasets/mitishaagarwal/patient.

Janitza, Silke, Ender Celik and Anne-Laure Boulesteix. "A Computationally Fast Variable Importance Test for Random Forests for High-Dimensional Data." Advances in Data Analysis and Classification, vol. 12, no. 4, 2016, pp. 885–915.

Degenhardt, Frauke, Stephan Seifert and Silke Szymczak. "Evaluation of Variable Selection Methods for Random Forests and OMICS Data Sets." Briefings in Bioinformatics, vol. 20, no. 2, 2017, pp. 492–503.

Kursa, Miron B., and Witold R. Rudnicki. "Feature Selection with the Boruta Package." Journal of Statistical Software, vol. 36, no. 11, 2010, pp. 1-13.

Genuer, Robin, Jean-Michel Poggi, Christine Tuleau-Malot. "VSURF: An R Package for Variable Selection Using Random Forests." The R Journal, R Foundation for Statistical Computing, vol. 7 no. 2, 2015, pp.19-33.

Genuer, Robin, Jean-Michel Poggi, Christine Tuleau-Malot. "Variable Selection Using Random Forests." Pattern Recognition Letters, vol. 31, no. 14, 2010, pp. 2225–2236.

Speiser, Jaime Lynn, Michael E. Miller, Janet Tooze and Edward Ip. "A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling." Expert Systems with Applications, vol. 134, 2019, pp. 93–101.

Díaz-Uriarte, Ramón, and Sara Alvarez de Andrés. "Gene Selection and Classification of Microarray Data Using Random Forest." BMC Bioinformatics, vol. 7, no. 1, 2006.

Darst, Burcu F., Kristen C. Malecki and Corinne D. Engelman. "Using Recursive Feature Elimination in Random Forest to Account for Correlated Variables in High Dimensional Data." BMC Genetics, vol. 19, no. S1, 2018.

Gregorutti, Baptiste, Bertrand Michel and Philippe Saint-Pierre. "Correlation and Variable Importance in Random Forests." Statistics and Computing, vol. 27, no. 3, 2016, pp. 659–678.

Bag, Souvik, Kapil Gupta and Soudeep Deb. "A review and recommendations on variable selection methods in regression models for binary data." Indian Institute of Management Bangalore, 2022.

Chen, Xi, and Hemant Ishwaran. "Random Forests for Genomic Data Analysis." Genomics, vol. 99, no. 6, 2012, pp. 323–329.

Breiman, Leo. "Random Forests." Machine Learning, vol.45, 2001, pp. 5-32.

Altmann, André, et al. "Permutation Importance: A Corrected Feature Importance Measure." Bioinformatics, vol. 26, no. 10, 2010, pp. 1340–1347.

Schonlau, Matthias, and Rosie Yuyan Zou. “The Random Forest Algorithm for Statistical Learning.” The Stata Journal: Promoting Communications on Statistics and Stata, vol. 20, no. 1, 2020, pp. 3–29.