Evaluating Class Imbalance and Asymmetric Costs Using Machine Learning

by

Samuel Meeks, B.A.

A Dissertation

In

Educational Psychology

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY

Approved

Todd Little
Chair of Committee

Jaehoon Lee

Eugene Wang

Mark Sheridan
Dean of the Graduate School

August, 2020

TABLE OF CONTENTS

ABSTRACT

The current study will evaluate the use of machine learning as a form of risk assessment as it fits within the risk-need-responsivity framework. Specifically, the risk of violent reconviction will attempt to be predicted by multiple machine learning algorithms. As violent reconviction has significant class imbalance, as well as asymmetric error cost, methodologies accounting for these potentially problematic situations will be evaluated.

While machine learning has been shown as an improvement over traditional assessment, more research is necessary to determine the most effective practices when applying its specialized methodologies. Analysis of the techniques used as treatment for class imbalance and asymmetric cost has not been researched on actual criminal justice data, leading to a gap in the scientific literature necessary to evaluate their genuine performance when applied.

LIST OF TABLES

LIST OF ABBREVIATIONS

AUC: Area Under the ROC Curve

CART: Classification And Regression Trees

CHANGES: Changing Habits and Achieving New Goals to Empower Success

CIP: Cognitive Intervention Program (CIP)

CTE: Career and Technical Education

HCR-20: Historical Clinical Risk Management-20

LSI-R: Level of Service Inventory-Revised

PCL-R: Psychopathy Checklist-Revised

RNR: Risk-Need-Responsivity

ROC Curve: Receiver Operating Characteristic Curve

SARA: Spousal Assault Risk Assessment

SAVRY: Structured Assessment of Violence Risk in Youth

SORAG: Sex Offender Risk Appraisal Guide

SMOTE: Synthetic Minority Over-sampling Technique

SVR-20: Sexual Violence Risk-20

TDCJ: Texas Department of Criminal Justice

VRAG: Violence Risk Appraisal Guide

CHAPTER 1

INTRODUCTION

Popular opinion on how to reduce criminal behaviors and their cost, both financial and societal, have varied over time. An emphasis on punishment and deterrence gained popularity during the 1970s and set precedents that persist to this day; however meta-analyses showing the effectiveness of rehabilitation and the development of a psychology of criminality caused a resurgence in rehabilitation efforts beginning in the late 1980s and early 1990s (Andrews & Bonta, 2010).

Effective rehabilitation should seek to reduce future criminal activity, thereby reducing future costs on national, state, and local governments due to re-incarcerating offenders. Reducing the severity of future criminal activity could also reduce the cost on society at a more personal level. Violent crime, while difficult to predict due to its lower prevalence rate, should be targeted specifically for rehabilitation. Violent crime creates fear in society, and the public's perception is that violent crime is worse than non-violent crime, evidenced by the fact that as of December 2019, 29 out of 50 states in the U.S. authorize capital punishment for extreme cases of violent crime (McInnes, 2019).

Risk-Need-Responsivity

Andrews, Bonta, and Hoge (1990) recognized the necessity for a psychological model of criminality and created the Risk-Need-Responsivity (RNR) model as a way of defining the principles they believe underlie the rehabilitation process. The RNR model revolves around the idea that criminogenic risk, or the risk to reoffend, can be assessed and measured. The principle of risk in the RNR model is that the severity of an offender's criminogenic risk should influence the intensity of rehabilitation interventions given.

Offenders with a sufficiently low criminogenic risk ideally would receive no intervention, while offenders with the highest criminogenic risk would receive the most intensive interventions. This not only helps direct where interventions will be most useful, but also helps save money and resources used by eliminating unnecessary interventions.

The nature of the potential crimes must be evaluated so criminal justice experts may effectively target their rehabilitation efforts to meet the needs of the offender. This is the principle of need, and since these criminogenic needs make up the risk of reoffending, offenders with higher risk expectedly have more criminogenic needs.

Finally, the principle of responsivity states that any intervention provided should be provided in a way that meets the learning style and abilities of the offender. Logically, for the interventions to be most successful, interventions should be offered in a way the offender can best understand.

Assessment In RNR

Naturally, criminal justice experts need a way to measure an incarcerated offender's risk of committing future crime in order to make decisions about that risk. This is traditionally performed via questionnaire-style assessments, where a risk level is designated based on the sum of the scored attributes questioned. Research has shown that measuring risk in this way tends to reach a ceiling on prediction that traditional assessment is unable to pass (Howard, 2017), while using machine learning as the methodology of assessment has been shown to break that ceiling (Ozkan, 2017; Curtis, 2018).

Machine learning as a form of risk assessment can easily fit within the Risk-Need-Responsivity model. Machine learning should be the first phase of assessment

performed, utilized to screen-out offenders with low enough risk to need no intervention, and identify high risk offenders for a second phase of assessment. Those identified as in need of additional assessment should be further evaluated on their criminogenic needs, and the machine learning assessment can help inform criminal justice experts on what criminogenic needs might require evaluation. Using machine learning in this way fulfills the risk and need principles of the RNR model by allowing criminal justice experts to appropriately allocate interventions based on criminogenic need. The second phase of assessment should also be used to assess the learning abilities and styles of offenders, such that interventions may be designed to meet the responsivity principle.

Class Imbalance

One issue contributing to the difficulty in predicting violent crime is how low its prevalence is compared to non-violent crime. This low base-rate problem, referred to as class imbalance, can even cause more sophisticated methodologies such as machine learning to fail. Consider a criminal justice dataset where only 3% of the cases met the condition of being convicted of a violent crime. In such a case, any model that predicts every offender as not being at risk for violent crime would in theory be accurate 97% of the time.

Considering that this highly skewed distribution of outcome classes is the starting point, or essentially the prior distribution given to the model, it is reasonable to imagine why the model can have an issue finding a more accurate way to classify cases. Fortunately, methodologies to account for class imbalance exist, and largely operate by changing the ratio of the outcome classes in the prior distribution with which the model learns. Giving the algorithm a distribution rebalanced in such ways is not practically

different than the case of weighting the outcome such that the minority class grants a larger weight in the model than the majority class (Berk, 2019).

Since the algorithm is learning from a distribution of outcome classes which does not represent the actual prevalence rate in the population, accuracy of the model will always be sacrificed. As demonstrated earlier, this is not necessarily a negative thing. Rather than focusing on model accuracy, receiver operating characteristic (ROC) curves and the area under the ROC curve (AUC) can evaluate global model performance in ways generalizable to other assessments while not being as influenced by class imbalance.

Asymmetric Cost

A second issue arises when using only accuracy as the measurement of model success, particularly with an outcome such as violent behavior: accuracy makes no distinction about the errors the model makes. In the case of predicting the risk of someone being violent, those false positive and false negative errors can carry grave consequences. Should an offender be predicted to be high risk of future violence, yet in reality this offender would never have been violent (a "false positive"), it is unfortunate that the offender would continue their sentence. It is unfortunate for the offender, as well as unfortunate in that the resources continuing to be spent on that offender could be better used elsewhere.

If the offender's situation were reversed however, such that the offender was predicted to be low risk of future violence, yet in reality the offender is released into society and commits a violent crime (a "false negative"), that seems to carry more severe ramifications. This issue of disproportional errors is known as the costs being "asymmetric," as the relative cost of the errors are not equal. Machine learning

algorithms can incorporate the relative cost of the errors during model construction, allowing the model to learn in a way that theoretically produces a more desirable distribution of errors.

Conveniently, rebalancing the prior distribution of outcome classes, such as done in the case of class imbalance, also can be used to influence the error distribution. As the prevalence of the minority class is comparatively greater in the rebalanced prior distribution, the model can more easily determine trends associated with predicting the minority class, presumably leading to fewer false negatives overall.

A third method for managing the distribution of errors is done after the model is created. In any prediction problem, some cutoff point(s), or threshold(s), must exist for determining how each probability range is labeled. Obviously, adjusting this cut-point will change how the errors are being classified; thus, adjusting the cut-point can be used to distribute errors in more desired ways.

CHAPTER 2

LITERATURE REVIEW

Public Safety

Yang, Wong, and Coid (2010) noted that violence and its control are "significant social, political, criminal justice, mental health, and international security issues..." (p. 740). Not only does violence, or the fear thereof, have an immediate effect on those being victimized, it can drastically alter the perceptions and behaviors of the general public. Ultimately, Yang and colleagues (2010) recommended using risk assessment and management as a means to potentially reduce violence; however, they went on to note that the prediction of future violence has been one of the most complex and controversial issues in the behavioral sciences. Statistically, predicting the likelihood of future violence and being able to effectively use this data remain a challenge for professionals in the field.

The importance of public safety is evident in the amount of money society spends on it. This is especially true in the United States. Local, state, and national governments invest billions of dollars every year in efforts to promote public safety (policing, criminal justice, border control, etc.). While these investments tend to range from 1.7% to 6.0% of each of the state's total revenue (M = 3.2%, SD = .87%), some states, like California, invested over $11 billion in 2016 (US Census Bureau). In Texas alone, the annual state budgets for the Texas Department of Criminal Justice, Texas Juvenile Justice Department, and Texas Department of Public Safety exceed $5 billion (US Census Bureau). When local (city and county) and federal expenditures are also considered, they are likely to add up to tens of billions of dollars in Texas alone.

Risk-Need-Responsivity

In an attempt to provide a theoretical framework for the rehabilitation process, Andrews, Bonta, and Hoge (1990) proposed the Risk-Need-Responsivity (RNR) model for assessing and treating criminogenic needs. Their focus was on creating a model that was practical in nature, providing effective service to those that will benefit most from them. Three principles were the foundation of the model: risk, need, and responsivity (Andrews and Bonta, 2010). Andrews and Bonta (2010) further found three factors important in predicting criminal behavior: biology/temperament, personality, and cognition, and explain the relationship by stating: "personal capabilities and predispositions affect how the environment influences the shaping of behavior and, reciprocally, behavior can modify biological tendencies" (pp. 159-160).

Risk. The principle of risk states interventions/services are to be consistent with the offender's likelihood to reoffend, such that more intensive interventions are to be used on the offenders identified with the highest criminogenic risk (Andrews and Bonta, 2010). In practice, criminogenic risk must be assessed so that interventions may be utilized effectively. Offenders with low risk should be identified as needing no intervention, saving potential resources that would have been used to treat them unnecessarily. Furthermore, this risk level must influence the intensity of the interventions to those who receive them to meet the principle of risk.

Psychopathy. Regardless of the methodology used, any assessment of risk will only be as predictable as the variables used. To help better predict violent behavior, more informative predictors will be needed. Research on the Revised Psychopathy Checklist-

revised (PCL-R) has shown that assessment of psychopathy is highly predictive of treatability, recidivism, and violence (Millon, Simonsen, Birket-Smith, & Davis, 2003).

The Revised Psychopathy Checklist is the most widely used validated measure for assessing psychopathy in forensic populations (PCL-R; Hare, 1991). Psychopathy has been shown to be related to a disproportional amount of serious repetitive crime and violence, and psychopaths are typically more criminally active across their lifespans than other offenders (Hare, 1991). Widiger and Trull (1994) suggested that psychopathy is a moderator variable that interacts with aggressiveness.

Serin and Amos (1995) found that in a three-year time window observing 299 male offenders, 40% of those previously classified as psychopaths were reconvicted for a violent crime while only 10% of nonpsychopaths were reconvicted for a violent crime in the same sample. Wintrup (1994) found in another study that over 60% of offenders with at least a 30 PCL-R score recidivated within 5 years while only about 20% of those with less than a 30 PCL-R score recidivated. Millon et al. (2003) noted that PCL-R scores are also predictive for female and adolescent offenders.

Need. The principle of need states that there should be an assessment of offender's criminogenic needs and those needs should be treated in order to reduce the likelihood of reoffending. Since offender's needs contribute to their criminogenic risk, higher risk offenders will likely have more needs (Andrews & Bonta, 2015). Effective treatment of the identified criminogenic needs will consequently reduce an offender's criminogenic risk. Criminogenic needs can change over time and across different incarcerations, thus establishing a necessity to reassess these needs over time.

Central Eight. During the same period the Risk-Need-Responsivity model was being developed, the first version of the Level of Service Inventory-Revised (LSI-R) was released (Andrews, 1982). The LSI-R was used to assess criminogenic risk and criminogenic need, and included these dimensions/categories: antisocial attitudes, antisocial associates, criminal history, substance abuse, family/marital, school/work, leisure recreation, financial problems, accommodation problems, or personal/emotional issues. Andrews and Bonta (2010) continued working on the LSI-R, developing it into what is now the Central Eight risk and need factors.

The Central Eight risk and need factors are comprised of the "Big Four" and the "Moderate Four" risk and need factors (Andrews, Bonta, & Wormith, 2006). The Big Four criminogenic risk/need factors are history of antisocial behavior, antisocial personality pattern, antisocial cognition, and antisocial associates. The Big Four are theoretically intercorrelated; however, Andrews et al. (2012) noted that their separate assessment allows for the identification of specific criminogenic needs to be used in case management and program planning. The Moderate Four, theoretically less strongly related to criminal activity, are family/marital circumstances, school/work, leisure/recreation, and substance abuse (Andrews et al., 2012).

Andrews and Bonta (2010) summarized eight meta-analytic reviews of the predictive validity of central eight risk/need factors on criminal recidivism stating that, "The grand mean r value for the big four was .26 (95% CI of .22/.39, k = 24) compared to .17 (.13/.20, k = 23) for the more modest four" (p. 116). Wang (1998) further demonstrated the predictability of antisocial personality patterns on physical aggression.

Responsivity. The principle of responsivity states that, to be successful, interventions should be match the offender's learning style, motivation, abilities, and strengths (Andrews & Bonta, 2007). It is necessary that these factors be assessed so that any and all interventions given are designed and implemented in a way that the offender can best benefit. While much research has been performed on the risk and need principles, responsivity remains understudied (Polaschek, 2012).

Traditional Criminal Risk Assessment

For much of the first half of the 20th century, the assessment of criminal risk was left in the hands of criminal justice staff (i.e., probation officers and prison staff) and clinical professionals (i.e., psychologists, psychiatrists, and social workers). Guided by their own professional training and experience, staff would make decisions as to who required enhanced security and supervision. Therefore, the assessment of risk was purely a matter of professional judgment (Bonta & Andrews, 2007).

Beginning in the 1970's there was a growing recognition that the assessment of risk needed to depend more upon actuarial, evidence-based science and less on professional judgment (Bonta & Andrews, 2007). This was motivated by Meehl's (1954) distinction between "clinical" and "actuarial" (or statistical) approaches. Meehl (1954) shaped much of the discussion of methods for combining information for applied decision making in psychology. Meehl defined actuarial predictions as having two features: "they use an explicit method of combining the information, and link this information to a probability figure on the basis of empirically determined relative frequencies".

The period between 1970 and 1980 saw a movement from what Bonta (1996) called “first generation” risk assessment (i.e., professional judgments of risk) to “second generation” risk assessment (i.e., actuarial assessment of risk). It became clear that these actuarial risk assessment instruments were better at predicting criminal behavior than professional judgment. Research reviews repeatedly showed that actuarial instruments performed better than clinical or professional judgment when making predictions of human behavior, across a variety of predictive tasks (Grove & Meehl, 1996).

A “third generation” of tools emerged in the late 1970s and early 1980s that combined dynamic factors with static factors. Examples of dynamic factors are present employment, criminal friends, family relationships, etc. Static factors, on the other hand, are stable traits (such as gender) as well as historical factors (such as criminal history). This third generation is referred to as "risk-need" instruments (Bonta & Andrews, 2007).

The introduction of “fourth generation” risk assessment instruments began in the early 1990s. These instruments claimed to integrate (a) systematic intervention and monitoring, (b) assessments of a broader range of offender risk factors not previously measured, and (c) other personal factors important to treatment (Andrews, Bonta, & Hoge, 1990). An example of a fourth generation risk assessment instrument is the Level of Service/Case Management Inventory (LS/CMI; Andrews, Bonta, & Wormith, 2004).

Singh, Grann, and Fazel (2011) identified over 120 risk assessment tools currently used in general and psychiatric settings. Of those 120+, the following nine have been studied the most: (1) Historical Clinical Risk Management-20 (HCR-20), (2) Level of Service Inventory-Revised (LSI-R), (3) Psychopathy Checklist-Revised (PCL-R), (4) Sex Offender Risk Appraisal Guide (SORAG), (5) Sexual Violence Risk-20 (SVR-20), (6)

Spousal Assault Risk Assessment (SARA), (7) Static-99, (8) Structured Assessment of Violence Risk in Youth (SAVRY), and (9) Violence Risk Appraisal Guide (VRAG).

Although commonly used, Yang et al. (2010) found that the predictive accuracy of these nine risk assessment instruments were essentially "interchangeable," with estimates of accuracy falling within a narrow band (Area Under the Curve = .65 to .71). They concluded: "The moderate level of predictive accuracy of these tools suggests that they should not be used solely for some criminal justice decision making that requires a very high level of accuracy such as preventive detention" (p. 740). Howard (2017) noted that some authors have ascribed a "glass ceiling" of predictive validity of criminal risk to an AUC level at or slightly above .70.

Risk assessment research can benefit from statistical methodological innovations found in the "machine learning" (also referred to as "statistical learning" or "predictive modeling") fields. Risk assessment is simply one type of predictive modeling task and can therefore take advantage of the methodological innovations these fields have developed.

Machine Learning

Since Monahan and Steadman (1994) listed the methodological problems which have plagued violence risk research, there have been tremendous strides made in statistical analyses/modeling. In 1994, most of the violence risk research was predicated on descriptive statistics (i.e., monotonically increasing recidivism rates for low/medium/high score groups), or parametric approaches based on the general linear model (e.g., linear regression) or the generalized linear model (e.g., logistic regression).

While generalized linear models had a role to play in predicting risk of violence, Berk, Sherman, Barnes, Kurtz, and Ahlman (2009) shifted the focus by suggesting that nonlinear and contingent relationships between predictors and outcomes could not be as easily modeled using traditional approaches. Indeed, Berk et al. (2009) claimed that there are a great deal of non-linear associations that, at that time, had yet to be incorporated into theory. They ultimately concluded that "...non-linear relationships also help to explain why random-forests forecasts do so much better than logistic regression and other parametric regression procedures. In this instance, the parametric regression models got the functional forms very wrong because the actual functional forms were unknown before the data analysis began" (pp. 205 - 206).

Thus, as Berk and Bleich (2013) pointed out, "complex decision boundaries pose a significant challenge for logistic regression or any parametric classifier. To forecast well, a researcher must understand the nature of the complexity, be able to translate that knowledge properly into an algebraic expression, and then have the data to construct an appropriate model. These requirements are daunting for criminal justice applications" (p. 541).

Since the mid-1990's, statistical modeling (particularly tree-based modeling) has advanced by leaps and bounds, and the risk assessment field can utilize these newer statistical methods to improve prediction accuracy. However, these new methods have not permeated the risk assessment field to any large degree. As Berk and Bleich (2013) pointed out, "there can be a substantial disconnect between the technical literature and the applications favored by many criminal justice researchers" (p. 515).

Beginning with Gardner, Lidz, Mulvey, and Shaw (1996), criminal justice researchers began utilizing classification and regression trees (initially referred to as CART; however, CART™ has since been trademarked). Of the benefits in using classification and regression trees, Berk and Bleich (2013) noted that they (1) have been established using rigorous mathematics, as well as a well-justified rationale and years of evaluation on real data; (2) construct accurate profiles of individuals associated with different outcome classes (i.e., categorical, polytomized) by subsetting the data into groups of people with similar profiles; (3) can account for false positives and false negatives in the training and prediction algorithms; (4) automatically incorporates step and hierarchical functions and interactions; and (5) can be used to further understand a subject-matter via accurate classification of individuals with similar profiles on an intended outcome. Due to these advantages over traditional parametric analyses, several researchers used single classification or regression trees (Gardner et al., 1996; Parent, Guay, & Knight, 2012; Rosenfeld & Lewis, 2005; Stalans, Yarnold, Seng, Olson, & Repp, 2004) or extensions labeled "iterative classification trees" (Banks et al., 2004; Monahan et al., 2005; Silver & Chow-Martin, 2002; Skeem et al., 2004; Steadman et al., 2000).

Unfortunately, results from single classification or regression trees can be unstable. With new data, such as those one would use to make forecasts, predictive accuracy can be disappointing (Berk & Bleich, 2013). James, Witten, Hastie, and Tibshirani (2013) noted that although single tree-based methods are simple and useful for interpretation, they typically are not competitive with the more accurate “ensemble” algorithms such as bagging, random forests, or boosting. Each of these approaches

involves producing multiple trees which are then combined to yield a single consensus prediction. Combining a large number of trees can often result in dramatic improvements in prediction accuracy, but at the expense of loss in interpretability.

Berk, Bleich, and their colleagues (2009, 2013) have been ardent advocates for the use of machine learning approaches in criminal justice research that aims at prediction (also referred to as "forecasting" or "discrimination") tasks. Berk and Bleich (2013) stated: "The transition to machine learning can confer a number of important benefits, some of which are not available otherwise. First, one is not limited to classifiers able to forecast one of two outcome categories. Second, forecasting errors that do not have equal costs can be introduced into the procedure at the beginning so that all of the results properly represent the preferences of stakeholders. Third, regularization is often built directly into the procedure to increase forecasting accuracy. Fourth, highly unbalanced distributions for the classes to be classified create no special problem as long as rare outcomes are important enough to be given extra weight in the analysis. Finally, some procedures will work well and in a principled manner, even when a very large number of predictors is included in the analysis." (p. 527).

Berk and Bleich (2013) concluded: "Random forests and stochastic gradient boosting represent true machine learning procedures based on ensembles of classification trees. Both are nonparametric, rest on solid mathematical foundations, and have been widely battle tested. All of the evidence to date indicates that they can perform well in criminal justice applications" (p. 527).

Ozkan (2017) used multiple machine learning algorithms to try to predict recidivism in a national sample of prisoners released in 1994. Although it was difficult to

determine exactly how many variables and what exact sample size he ultimately used in his study, he compared logistic regression, random forests, support vector machines, XGBoost, neural networks, and the Search algorithm. He found that XGBoost had the best predictive performance (test sample AUC = .824)--substantially higher than the .70 "glass ceiling" that Howard (2017) noted of traditional risk assessment tools.

Curtis (2018) used a sample of 258,248 offenders released from prison between 2010 and 2013. He used 109 predictors and constructed 10 predictive models using random forests and XGBoost algorithms across five different outcomes (all at three years post-release): recidivism (i.e., reincarceration), rearrest for any offense, reconviction for any offense, rearrest for a violent offense, and reconviction for a violent offense. He used 5-fold cross-validation to identify optimal tuning parameters, and cross-validated his training model on a separate (20%) test sample.

Curtis (2018) found that random forests had marginally better AUC than did XGBoost across all five outcomes; however, the two algorithms did not differ on overall accuracy, sensitivity, or specificity. For recidivism, random forests and XGBoost had test sample AUC values of .85 and .83, respectively. Thus, Curtis' (2018) overall model performance of the random forests and XGBoost algorithms were almost exactly what Ozkan (2017) found for recidivism, although with a very different sample and set of predictors.

Curtis (2018) further found that several variables were important to all 10 models. Three showed up in the top 20 most important variables in all 10 variable important lists: age, number of tattoos, and good time lost from disciplinary incidents during the most recent incarceration. Three more showed up in the top 20 of 9 of the 10 models: number

of times convicted of a property offense, minor disciplinary incidents during the current incarceration, and age at first arrest. Number of prior convictions (for any offense) also showed up frequently in the variable importance lists.

Thus, Ozkan's (2017) and Curtis' (2018) findings established the performance improvement of using random forests and XGBoost machine learning algorithms in the predictive accuracy of recidivism. The AUC's found by Ozkan (2017) and Curtis (2018) were approximately 18%-20% higher than the .70 "glass ceiling" that Howard (2017) noted.

AUC

A simple and commonly used metric for measuring a predictive model's performance is the overall accuracy rate. However, Kuhn and Johnson (2013) noted a few disadvantages to using this statistic. First, overall accuracy counts make no distinction about the of errors being made, and in situations where the costs of predictive errors are different, accuracy may not measure the important model characteristics. Second, one must consider the natural frequencies, or base rate, of each class.

With extremely low base rate outcomes, it is easy to imagine the possible issues with using accuracy as the measure of performance. Consider an outcome where the class of interest is the minority class, and only observed in three percent of participants in a dataset. A model that predicts everyone as belonging to the majority class would be accurate 97% of the time. While this metric confirms that the model is correct in its predictions almost every time, it is not helpful in predicting the minority class of interest. Every observation in the minority class would be predicted incorrectly, and those false predictions can have serious consequences in practice. Should the minority class be the

existence of cancer cells within a patient, the complete misclassification is obviously not acceptable no matter how accurate the model is at predicting the absence of cancer cells.

Rather than use overall accuracy, other metrics of global model performance have been adopted, primarily the receiver operating characteristic (ROC) curve and the area under the ROC curve (AUC). Howard (2017) noted that the AUC has become a key measure of the predictive validity of risk assessment instruments, and Helmus and Babchishin (2017) noted that AUC is still the most commonly used and recommended statistic for risk assessment scales.

When predicting two classes such as violent reconviction, there must exist a cut-point (e.g., threshold) which differentiates assignment of class prediction. The typical default cut-point is a 50% probability, such that cases with a probability over 50% of being in the predicted class are assigned to the predicted class by the model. There exist an infinite number of cut-points in the range of 0 – 100%; each cut-point will produce different proportions of false positive and false negative errors. An ROC curve is the graphical representation of every cut-point, and the tradeoff in false positive and false negative rates across those cut-points. AUC, previously mentioned as being the area under the ROC curve, represents the overall predictability across all cut-points. The maximum AUC value of 1 represents prefect predictability, while the AUC value of 0.5 represents predictability equivalent to chance, no better than flipping a coin.

In criminal justice practice, Tollenaar and Van der Heijden (2013) reported that an AUC value can also be interpreted as the probability that a randomly selected recidivist would have a higher score (or probability) than a randomly selected nonrecidivist. This interpretation of course extends into any other binary outcome, such

as reconviction. Helmus and Babchishin (2017) noted two advantages of AUC for risk assessment are that AUC is not unduly affected by the base rate of recidivism and can also be interpreted in the same way regardless of the measurement scale of the predictor. See *Table 1* for descriptive labels of AUC ranges suggested by Tollenaar and Van der Heijden (2013).

Table 1: Tollenaar & Van der Heijden's AUC Classifications

AUC	Classification
0.50	Chance
0.70	Acceptable
0.75	Large
0.80	Excellent
0.90+	Outstanding

The AUC is not without its limitations. For example, Mossman (2013) noted: "these indices do not provide details about sensitivity-specificity trade-offs; they do not tell us how to balance false-positive and false-negative errors; and they do not determine whether a diagnostic system is accurate enough to make practically useful distinctions between violent and non-violent subject groups. Justifying choices or clinical practices requires a contextual investigation of outcomes, a process that takes us beyond simply knowing global indices of accuracy" (p. 37).

Instead of AUC, the ROC curve can be used to make judgements about how to balance the errors. Howard (2017) also noted two sources of variation in general and violent recidivism AUCs: heterogeneity in risk predictor scores related to offender characteristics, and grouping of risk scores into categories. See Fawcett (2006) and Singh (2013) for technical aspects of ROC curves and AUC.

Sensitivity and Specificity

For two class predictions, such as with the outcome violent reconviction, there are additional statistics that may be relevant when one class is interpreted as the event of interest (e.g., violent behavior). The sensitivity, or true positive rate, of the model is the rate that the event of interest is predicted correctly for all individuals having the event. Conversely, specificity, or the true negative rate, is the rate that the event of interest is correctly predicted not to occur for all individuals for whom the event truly does not occur. Potential trade-offs exist between sensitivity and specificity when there are different penalties or costs associated with each type of predictive error (e.g., false positives vs. false negatives).

Fortunately, in addition to being a global measure of model performance, the ROC curve is a helpful tool for choosing a cut-point that appropriately maximizes the trade-off between sensitivity, specificity, and the errors. Given a set of continuous data points, the ROC curve can help determine an effective cut-point such that values above the threshold are indicative of a specific event [such as violence] (Kuhn & Johnson, 2013).

Several techniques exist for determining a new cut-point: (1) a particular target; (2) find the top left point on the ROC curve; (3) based on a ratio of false positives and false negatives, and (4) Youden's Index (Youden, 1950). Youden's Index balances the tradeoff between sensitivity and specificity by identifying the probability value which maximizes the sum of sensitivity and specificity (Youden, 1950). The cut-point associated with the largest Youden's index may show superior performance relative to the default 50% value (Kuhn & Johnson, 2013).

Thus, to be consistent with prior research, AUC will be used as a general measure of model performance. The sensitivity/specificity tradeoff will be evaluated by using Youden's Index and ratios of false positive and false negatives.

Asymmetric Cost

When false positive and false negative error are considered unequal and one is desired more or less than the other, one is said to be working with asymmetric costs. In his *Machine Learning Risk Assessments in Criminal Justice Settings,* Berk explains the model's votes need to be weighted by their relative cost, else "…the forecasts are not likely to be responsive to the needs of the decision makers, who can be badly misled." (2019, p. 51). Furthermore, Berk suggested that building the relative costs of the error at the beginning of the procedure can be beneficial when predicting highly differing costs (2019, p. 78).

Three main approaches have shown success in compensating for asymmetric cost. One approach, often called cost-sensitive learning, is to weight the training and test data by the relative costs as a parameter during the model training and tuning (Berk, 2019). A second is to influence directly the outcome classes in the prior distribution, such as by case weighting or changing the ratio of cases to reflect those costs (Berk, 2019). The third approach is done during the model post-processing, and involves moving the cut-point of prediction to balance the ratio of false positives to false negatives (Kuhn & Johnson, 2013).

Class Imbalance

For the purpose of this study, the problem of class imbalance refers specifically to the negative impact highly disproportional outcome classes, such as with a low base-rate,

can have on prediction. As a machine learning model is being trained on a highly disproportional outcome, the majority class examples can overwhelm the model (Thai-Nghe, 2010). Put simply, the model will more easily and accurately predict the majority class while sacrificing the predictability of the harder to predict minority class.

This working definition is necessary, as the problem of class imbalance is so often prevalent in data with asymmetric cost that some researcher consider them an interchangeable issue. Eitrich et al. define "…a data set to be unbalanced if either the sizes of the two classes differ significantly, or the costs for a false negative classification are very high whereas a false positive is acceptable, or if both conditions hold" (2007, p. 92). The authors then demonstrated success with cost-sensitive learning, rebalancing the prior distribution, and cut-points used in combination to classify a highly unbalanced pharmaceutical dataset (Eitrich, 2007). Thai-Nghe et al. (2010) further demonstrated the versatility of cost-sensitive learning and rebalancing the prior, finding the combination of both reduced the misclassification cost in 17 of 18 imbalanced datasets. Zhou et al. (2006) found cost-sensitive learning to be easily applied successfully to two-class outcomes, and demonstrated the effectiveness of cut-point moving in conjunction with cost-sensitive learning. Finally, cost-sensitive learning has even shown success on highly imbalanced outcomes including a 2.5% base-rate, though some algorithms may perform better than others with highly disproportionate cost (Zadrozny, 2003).

The Current Study

The current study will evaluate the use of machine learning as a form of risk assessment as it fits within the risk-need-responsivity framework. Specifically, the risk of violent reconviction will attempt to be predicted by multiple machine learning

algorithms. As violent reconviction has significant class imbalance, as well as asymmetric error cost, methodologies accounting for these potentially problematic situations will be evaluated.

While machine learning has been shown as an improvement over traditional assessment, more research is necessary to determine the most effective practices when applying its specialized methodologies. Analysis of the techniques used as treatment for class imbalance and asymmetric cost has not been researched on actual criminal justice data, leading to a gap in the scientific literature necessary to evaluate their genuine performance when applied.

CHAPTER 3

METHODS

Participants

Participants will include 251,170 offenders released from the Texas Department of Criminal Justice (TDCJ) between the fiscal years 2010 and 2013 (i.e., September 1st 2010 – August 31st 2013). If an offender was incarcerated multiple times during this timeframe, only their first incarceration will be used. The data will be a sample of convenience, comprised of information already obtained by criminal justice professionals.

Outcome Variable

Violent reconviction was chosen as the outcome of interest. Violent reconviction is defined as an offender being reconvicted of a violent offence, as categorized by the National Crime Information Center guidelines (Aftergood, 2008). The following are considered categories of violent crime: assault, terroristic threats, trafficking, homicide, indecency with a child, sexual assault, sexual assault against a child, robbery, other violent sexual offenses, and kidnapping. Violent reconviction will have a base-rate of 5.03%, making it an imbalanced outcome as well as having asymmetric cost.

Predictor Variables

An exhaustive list of demographic, criminal history, and education predictors will be used. Demographic predictors will include characteristics like sex, age, race, place of birth, and number of tattoos and scars. Criminal history predictors will include age at first arrest and number of times arrested and convicted for different types of offenses as well as details about the offender's current incarceration. Also included will be education predictors (years of education, GED attainment and correctional educational

participation). The correctional educational programs offered within the Texas Department of Criminal Justice (TDCJ) are academic, Career and Technical Education (CTE), the Cognitive Intervention Program (CIP), and Changing Habits and Achieving New Goals to Empower Success (CHANGES). The predictor list will include both static (e.g., gender and place of birth) and dynamic variables (e.g., correctional educational program participation). In total, there will be 115 predictors.

Class Imbalance

Algorithms. There are many different statistical algorithms used in machine learning and many more constantly developed and refined. While the mathematical processes that underlie how these algorithms categorize outcomes and utilize predictors are different, they all have a similar process in being developed.

The development of a model through machine learning starts with selecting an algorithm to use. An example of a machine learning algorithm that will be used later is C5.0. When the model is actually being built, the processes is automatically performed by the algorithm. The second part of the machine learning process is choosing tuning parameters (or "hyperparameters") that will determine how the algorithm builds the model. The number of iterations the algorithm makes through the provided dataset and maximum depth of the model are common tuning parameters.

Finally, with the tuning parameters set and variables chosen, the model will be trained on a 80% subset of the total dataset and tested on 20% of the total dataset. The model is automatically built (e.g. trained) based on the dataset and parameters provided. This is where the bulk of the computation happens, and the way in which the model is created is mathematically different for each algorithm, allowing some to perform better

or worse than others. Cross-validation using 10 folds will be utilized when able during tuning to evaluate the performance of the parameters. The differences in algorithms and an explanation of their parameters are given below.

The R package Ranger implements the random forest algorithm in the C++ backend of R, resulting in a faster and more memory efficient random forest modeling (Wright & Ziegler, 2017). The algorithm recursively partitions an outcome into an ensemble of decision trees which then collectively weight each classification decision (Breiman, 2001). *Table 2* contains a list of parameters used for tuning in Ranger that will be specified. Mtry will be iterated from 16 to 20 through the trials. The split rule will be specified as gini, and the minimum node size will be held constant at 1.

Table 2: Tuning Parameters in Ranger

Parameter	Description
Mtry	Number of possible variables to use at each split.
Minimal Node Size	The minimum amount of cases needed to split into a new node.
Split Rule	Measure used for splitting. For classification and probability estimation: gini, extratrees, or Hellinger.

The R package "xgboost" will be used to perform "Extreme Gradient Boosting," a more efficient type of gradient boosting (Chen et al., 2019). *Table 3* contains a list of parameters used for tuning in xgboost that will be specified. Eta will use the following values as learning rates through the trials: 0.025, 0.05, 0.1, and 0.3. Gamma will be held constant at 0, and column sample by tree, minimum child weight, and subsample will all be held constant at 1. The maximum depth will iterated through the following values: 2, 3, 4, 5, and 6. Finally, the number of rounds attempted will iterate from 200 to 1,000 by intervals of 50.

Table 3: Tuning Parameters in Xgboost

Parameters	Description
Eta	The "learning rate", used to shrink each feature after boosting.
Gamma	Minimum loss reduction required to split.
Maximum Depth	Maximum depth a tree can grow.
Minimum Child Weight	Minimum sum of instance weight required to split.
Subsample	Ratio to subsample the training data.
Column Sample By Tree	Ratio to subsample for columns when making each tree.
Number of Rounds	Maximum number of boosting iterations.

C5.0 allows for two types of models to be developed: a decision-tree-based model, and a ruleset-based model. The C5.0 ruleset model will used in this study; a ruleset model produces easily interpretable if-then rulesets, allowing for the same dependencies as tree-based pathways, but also allow for pruning inside those dependencies. This means individual rules in a ruleset may be removed if found to be less informative to the outcome, even if they fall in the middle of the ruleset. *Table 4* contains a list of parameters used for tuning in C5.0 that will be specified. All models will be rules models that use winnowing, and trials will be held constant at 100.

Table 4: Tuning Parameters in C5.0

Parameters	Description
Trials	Number of boosting iterations.
Rules	Specify tree or rule model.
Winnow	Option to use winnowing (i.e., feature selection).

Rebalancing The Prior. Directly rebalancing the prior distribution of the outcome can help compensate for imbalanced outcomes, as well as asymmetric cost. One way to do this is by creating the training dataset in a unique method; sampling the original dataset so that the outcome classes are now represented more equally in the training dataset. Rebalancing the prior in this way is traditionally done by either up-

sampling such that the minority class is more represented, or down-sampling such that the majority class is less represented.

This study will use down-sampling to create its training dataset, taking the observations that meet the minority class (i.e., violent reconviction) from the original 80% partition, and sampling an equal number of observations that meet the majority class (i.e., no violent reconviction). Up-sampling will not be attempted, as a simulation by Kuhn (2019) has shown performance of down-sampling to be more stable. Furthermore, this study will utilize synthetic minority over-sampling technique (SMOTE), a more advanced method of rebalancing low base-rate outcomes (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

SMOTE uses a combination of up and down sampling, where the minority class is up-sampled by creating synthetic minority outcome classes using *k* nearest neighbors based on sampled cases. The majority class is down-sampled to a percentage of the minority. For example, if the majority class was under-sampled to 200%, the new dataset could contain twice as many observations of the minority class as the majority class.

Asymmetric Cost

Cost-Sensitive Learning. As previously mentioned, the errors of target outcomes are not always considered equal. Many machine learning algorithms can incorporate a researcher-specified cost ratio representing the ratio of weight to be placed on false positive and false negatives errors. This cost ratio is assigned to weight the classification of outcome classes during the model building. The exact way the algorithm incorporates the weighting during model building depends on the algorithm used, however, cost-sensitive learning specifically refers to weighting that occurs during the model building.

Cut-point Adjustment. As an alternative to cost matrices, asymmetric cost can be accounted for by adjusting the classification cut-point after the model has been created. This cut-point represents the probability an observation is classified into either violent reconviction or no violent reconviction. Most machine learning algorithms use a default value of 0.5, or a 50% probability threshold such that those with over a 50% probability are predicted as likely to be reconvicted of a violent offense.

This study will use the cutpointr package (Thiele, 2019) to calculate cut-points based on: a five-to-one ratio of false positives to false negatives, a ten-to-one ratio of false positives to false negatives, and the Youden's *J* Statistic (Youden, 1950). Adjusting the cut-point in this way allows the researcher to control where the errors are distributed.

Analyses

This study will evaluate the effect different combinations of algorithms and methodologies have on class imbalance and asymmetric cost. Each algorithm may perform differently, and some methods can compensate for both class imbalance and asymmetric cost, making many combinations necessary to evaluate. *Table 5* lists in rows each methodology to be evaluated, with columns displaying if the method should theoretically treat the issue of class imbalance or asymmetric cost. Each algorithm will be used in combination with each methodology to evaluate all possible combinations and their performance. Area under the curve will be used as an outcome for global model performance and is of interest when evaluating how algorithms respond to methods used to overcome class imbalance. Sensitivity and the number of false negatives will be evaluated as ways to determine the effect treatment for asymmetric cost has on the algorithms, as sensitivity represents the algorithms ability to predict the positive class and

false negative classification is the direct representation of how these methodologies affect the error of interest.

Table 5: Methodologies to be Evaluated

Method	Class Imbalance	Asymmetric Cost
Rebalance The Prior	Yes	Yes
Cost-Sensitive Learning	No	Yes
Cut-point Adjustment	No	Yes

CHAPTER 4

RESULTS

Rebalancing The Prior

Each algorithm was run without rebalancing the prior distribution. These models were first evaluated using the Youden cut-point, and then the cut-point adjustment of 5:1 and 10:1 was applied.

Both Random Forests and Extreme Gradient Boosting performed well without any rebalancing. *Table 6* shows the performance of each algorithm using no prior rebalancing as well as each cut-point used. C5.0 was unable to generate a model without rebalancing the prior, though as shown later, C5.0 was able to create a model without rebalancing the prior if cost-sensitive learning was used.

Table 6: No Rebalancing, No Cost-Sensitive

Model (Time)	AUC	Youden Cut-Point	5:1 Cut-Point	10:1 Cut-Point
C5.0	0.500	NA	NA	NA
Random Forest (32m)	0.827	Sens: 0.851, Spec: 0.667 FP:FN: 42.1:1 (15,880:377)	Sens: 0.337, Spec: 0.939 FP:FN: 1.8:1 (2,931:1,673)	Sens: 0.614, Spec: 0.832 FP:FN: 8.2:1 (7,997:974)
Extreme Gradient Boosting (6.5h)	0.858	Sens: 0.856, Spec: 0.719 FP:FN: 36.9:1 (13,395:363)	Sens: 0.428, Spec: 0.925 FP:FN: 2.48:1 (3,582:1,444)	Sens: 0.721, Spec: 0.821 FP:FN: 12.1:1 (8,546:704)

Cut-point adjustments of 5:1 and 10:1 ratios of false positive to false negatives did not really achieve their ratios precisely, though they were fairly close using the 10:1 ratio. The raw values for false positive and false negative classifications are listed underneath the ratios in parentheses. Compared to the Youden cut-point, the 5:1 and 10:1 adjustments had more false negative classifications in their attempts to create the desired

ratios. However, the 5:1 and 10:1 adjustments did have a large decrease in the number of false positive classifications, though this behavior was unexpected as the weight in cut-point adjustment was on false negative classifications. It is interesting that the cut-point adjustment prioritizes creating the desired ratio over weighting the false negative classifications effectively.

Cost-Sensitive Learning

Cost-sensitive learning was able to be applied to the C5.0 and Random Forest algorithms if they were used without any form of rebalancing. Two cost ratios were used when modeling with cost-sensitive learning: 5:1 and 10:1 ratios where false negatives were considered more costly than false positives. Using cost-sensitive learning in this way allowed C5.0 to create a model when it previously could not, even though it was not specifically expected to help with class imbalance. The C5.0 model did not perform as well as Random Forest regardless of metric used. Furthermore, due to a constraint in the C5.0 algorithm, only a single cut-point could be modeled, and it should be noted that the C5.0 cut-point used was not truly a Youden cut-point. This is due to C5.0 necessitating that new predictions use class outcomes (membership) rather than probabilities when measuring performance. See the C5.0 documentation for more details.

The results of using cost-sensitive learning using ratios of 5:1 and 10:1 are given in *Table 7* and *Table 8*, respectively. Random Forest using the Youden cut-point and cost-sensitive learning performed very similarly to Random Forest using Youden without cost-sensitive learning. There was not much difference between using a ratio of 10:1 or 5:1 for cost-sensitive modeling Random Forest. A major contrast occurs when applying the 5:1 and 10:1 cut-point adjustments to either of the cost-sensitive Random Forest

models. The adjustments both have the same results, the models predict only the positive class for all cases, resulting in a perfect sensitivity of one and specificity of zero. This interesting though undesirable behavior is only present when applying the 5:1 and 10:1 cut-point adjustments to cost sensitive learning models.

C5.0 was once again unable to generate a model when a cost-sensitive ratio of 5:1 was applied. Interestingly, when using a cost-sensitive ratio of 10:1 for C5.0, a model was able to be built. The C5.0 model using a 10:1 cost-sensitive ratio had very high specificity and had over double the number of false negative classifications than its Random Forest counterpart.

Table 7: 5:1 Cost-Sensitive Learning

Model (Time)	AUC	Youden Cut-Point	5:1 Cut-Point	10:1 Cut-Point
C5.0	0.500	NA	NA	NA
Random Forest (3.2h)	0.813	Sens: 0.809, Spec: 0.675 FP:FN: 32.2:1 (15,496:481)	Sens: 1, Spec: 0 FP:FN: (47,708:0)	Sens: 1, Spec: 0 FP:FN: (47,708:0)

Table 8: 10:1 Cost-Sensitive Learning

Model (Time)	AUC	Youden Cut-Point	5:1 Cut-Point	10:1 Cut-Point
C5.0 (11.5m)	0.652	Sens: 0.442, Spec: 0.863 FP:FN: 4.7:1 (6,552:1,408)	NA	NA
Random Forest (3.7h)	0.801	Sens: 0.787, Spec: 0.672 FP:FN: 29.1:1 (15,644:538)	Sens: 1, Spec: 0 FP:FN: (47,708:0)	Sens: 1, Spec: 0 FP:FN: (47,708:0)

Down-Sample Rebalanced Prior

All models were able to be implemented with down-sampling as the method of rebalancing the prior. Down-sampling allowed C5.0 to overcome class imbalance issues

and create a model where C5.0 without rebalancing the prior (and no cost-sensitive learning) could not.

Some patterns emerged across all models when using down-sampling as the method of rebalancing the prior. Sensitivity, specificity, and the ratio of false positive to false negative are very similar for each cut-point used regardless of algorithm. Even the raw number of classifications of false positives and false negatives are relatively similar across algorithms for each respective cut-point. When applying 5:1 and 10:1 cut-point adjustments on each model, the number of false negative classifications increase relative to the classifications from the Youden cut-point for all models, as was previously seen in the models using no rebalanced prior.

Compared to the models with no rebalancing of the prior and no cost-sensitive learning, the down-sampled models did not perform much differently with the exception of C5.0, which was previously unable to create a model. Both Random Forest and Extreme Gradient Boosting had very similar sensitivity, specificity, and false positive, and false negative classifications as their non-down-sampled counterparts across all cut-points.

Down-sampled C5.0 model using Youden seems to perform better than the 10:1 cost-sensitive learning model that was able to be produced (i.e., the Youden model) when looking at area under the curve as a global metric as well as its balance of sensitivity and specificity. The down-sampled C5.0 model had over double the amount of false positives but in turn had a better sensitivity, showing its tendency to predict the positive class. Full confusion matrices are listed in *Appendix A*. The cost-sensitive C5.0 model had over

three times as many false negative classifications even though it was trained to weight false negatives on a 10:1 ratio against false positives.

Random Forest down-sampled with the Youden cut-point performed very similar to both of the cost-sensitive Random Forest models, having just slightly less misclassifications in both the false positive and false negative categories. Down-sampled Random Forest did not have the issue using cut-point adjustment that cost-sensitive Random Forest did, where cost-sensitive Random Forest predicted only the positive class when applying cut-point adjustment.

Table 9: Down-Sampled Models

Model (Time)	AUC	Youden Cut-Point	5:1 Cut-Point	10:1 Cut-Point
C5.0 (24m)	0.850	Sens: 0.851, Spec: 0.715 FP:FN: 36.3:1 (13,603:375)	Sens: 0.455, Spec: 0.912 FP:FN: 3.1:1 (4,212:1,375)	Sens: 0.698, Spec: 0.822 FP:FN: 11.1:1 (8,472:763)
Random Forest (3.5m)	0.833	Sens: 0.817, Spec: 0.704 FP:FN: 30.6:1 (14,147:463)	Sens: 0.309, Spec: 0.946 FP:FN: 1.5:1 (2,588:1,744)	Sens: 0.657, Spec: 0.824 FP:FN: 9.7:1 (8,381:866)
Extreme Gradient Boosting (2.3h)	0.853	Sens: 0.853, Spec: 0.716 FP:FN: 36.5:1 (13,571:372)	Sens: 0.392, Spec: 0.930 FP:FN: 2.2:1 (3,350:1,535)	Sens: 0.706, Spec: 0.817 FP:FN: 11.8:1 (8,722:742)

SMOTE Rebalanced Prior

The synthetic minority over-sampling technique (SMOTE) was also used to rebalance the prior. C5.0 was able to produce a model using SMOTE where it could not without prior rebalancing. The results for models run with SMOTE are given below in *Table 10*. SMOTE rebalanced prior models yet again continued the trend of having more false negatives when using 5:1 and 10:1 adjusted cut-points for all models.

Using SMOTE with C5.0 had very similar results to C5.0 with down-sampling. For each cut-point used, C5.0 showed fairly similar sensitivity, specificity, and classifications of false positive and false negatives as their down-sampled counterparts. The C5.0 cost-sensitive learning model produced (i.e., the Youden cut-point model) had over four times the amount of false negative classifications and much worse sensitivity, though it had less than half the classifications of false positives than SMOTE C5.0 using Youden.

Random Forest using SMOTE to rebalance the prior performed similarly for each cut-point as its down-sampled counterparts, which in turn was not much different than Random Forest without any rebalancing (and no cost-sensitive learning). Rebalancing the prior with SMOTE for Random Forest did not show much difference than the Random Forest models using cost-sensitive learning and the Youden cut-point. Similar to down-sampled Random Forest, Random Forest with SMOTE did not have an issue using cut-point adjustment like cost-sensitive Random Forest did.

Extreme Gradient Boosting using SMOTE did not show noticeable improvement over down-sampling or without rebalancing the prior. As has been the case for every single algorithm used regardless of method (or lack of) rebalancing the prior, Extreme Gradient Boosting with SMOTE had an increase of false negative classifications when using 5:1 and 10:1 cut-point adjustments compared to Youden, and a decrease in false positive classifications.

Table 10: SMOTE Rebalanced Prior

Model (Time)	AUC	Youden Cut-Point	5:1 Cut-Point	10:1 Cut-Point
C5.0 (4.8h)	0.847	Sens: 0.869, Spec: 0683 FP:FN: 45.7:1 (15,121:331)	Sens: 0.418, Spec: 0.924 FP:FN: 2.5:1 (3,641:1,468)	Sens: 0.643, Spec: 0.845 FP:FN: 8.2:1 (7,407:901)
Random Forest (1.8h)	0.827	Sens: 0.798, Spec: 0.711 FP:FN: 27.1:1 (13,788:509)	Sens: 0.317, Spec: 0.943 FP:FN: 1.6:1 (2,715:1,723)	Sens: 0.653, Spec: 0.823 FP:FN: 9.7:1 (8,467:876)
Extreme Gradient Boosting (8.2h)	0.856	Sens: 0.855, Spec: 0.720 FP:FN: 36.5:1 (13,384:367)	Sens: 0.420, Spec: 0.923 FP:FN: 2.5:1 (3,653:1,463)	Sens: 0.760, Spec: 0.797 FP:FN: 16:1 (9,689:606)

CHAPTER 5

CONCLUSION

Class Imbalance

Of all the methodologies tested, only rebalancing the prior was expected to help algorithms overcome issues of class imbalance. The down-sampling and synthetic minority over-sampling technique (SMOTE) methods were used to compare with models without rebalancing the prior.

When using algorithms without rebalancing the prior, Random Forest and Extreme Gradient Boosting were able to produce a model. Looking at area under the curve (AUC) values as global metrics of model performance, the Random Forest model without any rebalancing (AUC = 0.827) performed almost as well as the Random Forest model with down-sampling (AUC = 0.833) and the same as Random Forest using SMOTE (AUC = 0.827).

Interestingly, Extreme Gradient Boost without rebalancing performed better (AUC = 0.858) than both the down-sampled model (AUC = 0.853) and the SMOTE model (AUC = 0.856), though the differences among the models are small.

Both down-sampling and SMOTE showed promise in overcoming class imbalance with C5.0. Using C5.0 without any rebalancing of the prior or built in cost-sensitive learning was unable to produce a model at all. When using either down-sampling or SMOTE to rebalance the prior however, C5.0 was able produce a model. Both the model using down-sampling (AUC = 0.850) and SMOTE (AUC = 0.847) with C5.0 performed better globally than their Random Forest counterpart using the same

rebalancing method, though not as well as its Extreme Gradient Boosting counterpart. Again, these are relatively minor differences.

These results suggest the algorithm used when predicting class imbalanced outcomes is more influential than simply using methods for rebalancing the prior indiscriminately. Random Forest when producing its model uses a type of bootstrap aggregation (bagging) when creating an ensemble of uncorrelated decision trees that make up the overall Random Forest model. This likely explains why Random Forest is able to produce a model without rebalancing the prior where C5.0 is not. Furthermore, Extreme Gradient Boosting is a gradient boosting algorithm. This means Extreme Gradient Boosting creates its ensemble by creating new models that fit new predictors to the residual errors made by previous iterations in the ensemble. Extreme Gradient Boosting not only performed best without any rebalancing of its prior than any other model by global metrics, Extreme Gradient Boosting also performed better using down-sampling and SMOTE than any other algorithm using the same rebalancing technique.

Gradient Boosting algorithms clearly show the most promise when dealing with class imbalanced data. If the situation necessitates the use of traditional algorithms such as C5.0, down-sampling or SMOTE can be used to overcome issues with imbalanced classes; however, Random Forest and Extreme Gradient Boosting would seem to be preferred because they do not need rebalancing to perform well. Down-sampling and SMOTE seem approximately equivalent in terms of performance, with down-sampling performing slightly better for C5.0 and Random Forest, and SMOTE performing slightly better for Extreme Gradient Boosting. If one of these methodologies must be used to rebalance the prior, researchers might consider the increased computational intensity and

time required for SMOTE compared to down-sampling as more of a determining factor than any minor difference in model performance.

Finally, it was unexpected to find that the 10:1 cost ratio for C5.0 using cost-sensitive learning was able to produce a model even without any rebalancing of the prior. It had by far the lower global performance (AUC = 0.652) and poor sensitivity (0.442), suggesting cost-sensitive learning with C5.0 should not be used to overcome class imbalance.

Asymmetric Cost

Three methodologies were evaluated as ways to handle asymmetric cost: rebalancing the prior, cost-sensitive learning, and cut-point adjustment.

Rebalancing the prior. Down-sampling as the method of rebalancing the prior did not show any improvement compared to no rebalancing of the prior. Comparing Random Forest and Extreme Gradient Boosting using down-sampling to their non-rebalanced counterparts, each model (across all cut-points) had more false negative classifications and lower sensitivity than not using any rebalancing for the prior. The only exception is that Random Forest, adjusted with a 10:1 cut-point ratio after using down-sampling, had a slightly higher sensitivity. This behavior is opposite of what was expected. The positive class was rebalanced using down-sampling so that it would be more prominent in the data than it was without down-sampling. It was expected that this would lead to models with higher sensitivity and less false negatives, as the classification of the positive class would be easier.

Rebalancing the prior using SMOTE showed very similar poor performance in terms of sensitivity and classifications of false negatives. Random Forest using SMOTE

had worse sensitivity for each respective cut-point except the 10:1 adjustment, and more false negatives for each respective cut-point except 5:1 as compared to down-sampling and no rebalancing used. Extreme Gradient Boosting had almost exactly the same sensitivity and classifications of false negatives when comparing SMOTE and no rebalancing with their Youden cut-points. Adjusting the cut-point to reflect a 5:1 ratio of false positives to false negatives increased the number of false negatives and decreased sensitivity compared to the same cut-point without rebalancing. Only when applying the 10:1 ratio for the cut-point did SMOTE show minor improvement over no rebalancing for sensitivity and false negatives.

C5.0 benefited the most from the use of down-sampling and SMOTE. Unfortunately, since C5.0 was unable to be modeled without resampling, there is not really a baseline model to compare against. However, C5.0 using down-sampling performed better than Random Forest using down-sampling, in terms of sensitivity and false negative classifications. C5.0 was also about equivalent to Extreme Gradient Boosting when using the Youden cut-point with down-sampling, and the two models had the two fewest false negative classifications and the two highest sensitivity values.

When considering the use of SMOTE, C5.0 had the least number of false negative classifications and the highest sensitivity of all algorithms using SMOTE when using the Youden cut-point. Extreme Gradient Boosting did have a performance similar to the C5.0 Youden model and had fewer false negatives and more sensitivity when using the 5:1 and 10:1 cut-point adjustments.

It seems that the algorithm is more influential to the classifications of false negatives as well as sensitivity than rebalancing the prior alone. When looking for the

highest sensitivity and the fewest false negatives, Extreme Gradient Boosting without rebalancing the prior and using the Youden cut-point outperforms all other models. Down-sampling and SMOTE should be a consideration when using traditional algorithms such as C5.0, yet SMOTE had only a minor increase in performance over down-sampling C5.0 at the cost of much more computational time building the model.

Cost-Sensitive Learning. Cost-sensitive learning models were able to be implemented without rebalancing the prior. Only C5.0 and Random Forest allowed for cost-sensitive learning, and C5.0 was limited to only using a single cut-point due to how the C5.0 package in R handles the classification of new cases with cost-sensitive C5.0 models. Furthermore, when attempting to apply 5:1 and 10:1 cut-point adjustments to the Random Forest models, they over compensated and only predicted the positive class for all cases.

C5.0 using the 5:1 weighting for false negatives to false positives was unable to overcome its issue with class imbalance and was unable to be modeled. When using the 10:1 weighting, C5.0 was able to be modeled, though its performance was not as good as C5.0 with down-sampling or SMOTE using Youden for AUC, sensitivity, or the classification of false negatives.

Random Forest inexplicably had greater sensitivity and fewer false negative classifications when using a weight of 5:1 for false negatives and false positives than when using the 10:1 weight that should weight false negatives more than the 5:1 model. This 5:1 Random Forest model performed slightly better on sensitivity and classification of false negatives than SMOTE using Youden, though not as well as its down-sampled and non-rebalanced counterparts. Cost-sensitive learning does not seem overall to be very

beneficial, and its erratic behavior when applying cut-point adjustments as well as its unpredictable results when applying different weighting makes using cost-sensitive learning much less consistent than any other method evaluated in this study.

Cut-point Adjustment. Three cut-points were utilized for each model for each method (and lack) of rebalancing. The Youden cut-point maximizes the sum of sensitivity and specificity in order to balance the tradeoff. The two other cut-points were calculated by using a weight ratio of false negatives to false positives. Ratios of 5:1 and 10:1 were used to increase the weight given to false negative classifications over false positive classifications. Given the results explained in the previous section on cost-sensitive learning, the attempts to use 5:1 and 10:1 adjusted cut-points on cost-sensitive models will not be elaborated on here.

Very clear trends emerged when comparing Youden, 5:1, and 10:1 cut-points. Youden had higher sensitivity and fewer false negatives for every model compared to 5:1 and 10:1 adjustments for their given method of rebalancing the prior. Using 5:1 and 10:1 adjustments consistently increased the number of false negative classifications while decreasing the number of false positive classifications in an attempt to achieve their desired ratios. The 10:1 adjustment had the expected result of having fewer false negatives than the 5:1 ratio; however, in all cases it produced more false negatives than the Youden cut-point.

As mentioned previously, the 5:1 and 10:1 cut-points both decreased the number of false positive classifications as compared to Youden. This also had the effect of decreasing the sensitivity compared to their Youden counterparts in all cases. While having more misclassifications of the positive class, Youden also had more true

classifications of the positive class than the 5:1 and 10:1 cut-points in all cases. Full confusion matrices are given in Appendix A. The only time using 5:1 or 10:1 had more true positive classifications were using Random Forest with cost-sensitive learning, as these models only predicted the positive class, thus predicting every positive class correctly and every negative class incorrectly.

It is very clear that using a 5:1 or 10:1 cut-point ratio of false negative to false positive is detrimental to the sensitivity, true positive classifications, and false negative classifications, having the exact opposite effect that these ratios were attempting to achieve. This suggests that researchers should prefer Youden's cutoff in any situation in an attempt to manage asymmetric error cost.

Recommended Practices

The choice of algorithm to use seems more influential and consistent than any other single factor for both class imbalanced data and asymmetric errors. When attempting to predict class imbalanced outcomes, Extreme Gradient Boosting shows the best performance and rebalancing the prior seems unnecessary. Random Forest is not too far behind Extreme Gradient Boosting, however. Rebalancing the prior through either down-sampling or SMOTE seem to be effective methods at overcoming issues with class imbalance in less sophisticated algorithms such as C5.0, and the performance in such cases are comparable to results shown from Random Forest and Extreme Gradient Boosting. The difference in down-sampling and SMOTE is negligible, and researchers should use down-sampling instead of SMOTE due to significantly less computational time needed.

Focusing on asymmetric error cost, cost-sensitive learning did not live up to expectations and using cut-point adjustments of 5:1 or 10:1 were simply not as effective as using Youden. C5.0 using SMOTE and the Youden cut-point had the most sensitivity and the fewest classifications of false negatives, though Extreme Gradient Boosting without rebalancing the prior and Youden was not far behind and had better global performance with slightly fewer false positives. Given the differences are minor, and SMOTE did not show any consistent improvement across all models while Extreme Gradient Boost was pretty consistently superior compared to other algorithms for each methodology evaluated, it is probably safest to use Extreme Gradient Boosting unless researchers have specific information or limitations that demand otherwise.

Limitations and Future Direction

Evaluation of cost-sensitive learning was somewhat limited by technical limitations and could be expanded on by future research. Only C5.0 and Random Forest were able to be modeled using cost-sensitive learning, and C5.0 was able to use only one cut-point. It would be interesting to see how other algorithms perform with cost-sensitive learning, particularly in the case of more traditional algorithms, since C5.0 was specifically limited.

Considering how well Extreme Gradient Boosting performed, it would be interesting to see how other boosting algorithms perform. The data used in this paper also had an outcome base rate of just under 6% for the positive class, and it would be beneficial for future research to evaluate the effect different proportions of the outcome class have on performance.

Finally, as alluded to in the literature review chapter of this paper, better predictors can have a profound effect on the modeling process. The data used in the paper was a sample gathered by the Texas Department of Criminal Justice for their own purposes and predictors were not able to be added by the researcher. Items indicative of violence such as the Revised Psychopathy Checklist-revised (PCL-R) which has been shown to be predictive of treatability, recidivism, and violence (Millon, Simonsen, Birket-Smith, & Davis, 2003) could lead to more accurate predictions of the imbalanced outcome class violence reconviction.

REFERENCES

Aftergood, S. (2008, June). National Crime Information Center (NCIC). Retrieved December 2019, from https://fas.org/irp/agency/doj/fbi/is/ncic.htm.

Andrews, D. A. (1982). *The level of supervisory inventory (LSI): 1. The first follow-up.* Ontario Ministry of Correctional Services, Planning and Support Services Division, Planning and Research Branch.

Andrews, D. A., & Bonta, J. (2010). Rehabilitating criminal justice policy and practice. *Psychology, Public Policy, and Law, 16*(1), 39–55. doi: 10.1037/a0018362

Andrews, DA & Bonta, James & Hoge, Robert. (1990). Classification for Effective Rehabilitation: Rediscovering Psychology. *Criminal Justice and Behavior. 17*. 19-52. 10.1177/0093854890017001004.

Andrews, D. A., Bonta, J., & Wormith, J. (2004). *The Level of Service/Case Management Inventory*. 10.1037/t05029-000.

Andrews, D. A., Bonta, J., & Wormith, J. S. (2006). *The recent past and near future of risk and/or need assessment*. Crime & Delinquency, 52(1), 7-27.

Andrews, D. A., Guzzo, L., Raynor, P., Rowe, R. C., Rettinger, L. J., Brews, A., & Wormith, J. S. (2012). *Are the Major Risk/Need Factors Predictive of Both Female and Male Reoffending?* International Journal of Offender Therapy and Comparative Criminology, 56(1), 113–133. doi: 10.1177/0306624x10395716

Banks, S., Robbins, P. C., Silver, E., Vesselinov, R., Stead-man, H. J., Monahan, J. ... Roth, L. H. (2004). A multiple-models approach to violence risk assessment among people with mental disorder. *Criminal Justice and Behavior, 31*(3), 324–340.

Berk, R. A. (2019). *Machine Learning Risk Assessments in Criminal Justice Settings*. Springer.

Berk, R. A. & Bleich, J. (2013). Statistical procedures for forecasting criminal behavior. *Criminology & Public Policy, 12*(3), 513–544.

Berk, R. A., Sherman, L., Barnes, G., Kurtz, E., & Ahlman, L. (2009). Forecasting murder within a population of probationers and parolees: A high stakes application of statistical learning. *Journal of the Royal Statistical Society: Series A (Statistics in Society), 172*(1), 191–211.

Bonta, J. (1996). Risk-needs assessment and treatment. In A. T. Harland (Ed.), *Choosing correctional options that work: Defining the demand and evaluating the supply* (p. 18–32). Sage Publications, Inc.

Bonta, J., & Andrews, D. A. (2007). *Risk-need-responsivity model for offender assessment and treatment*. User Report.

Breiman L (2001). "Random Forests." Machine Learning, 45(1), 5–32.

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

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., & Li, Y. (2019). Xgboost: Extreme Gradient Boosting. R package version 0.90.0.2. https://CRAN.R-project.org/package=xgboost

Curtis, J. (2018). On using machine learning to predict recidivism (Doctoral dissertation).

Eitrich, T., Kless, A., Druska, C., Meyer, W., & Grotendorst, J. (2007). Classification of Highly Unbalanced CYP450 Data of Drugs Using Cost Sensitive Machine Learning Techniques. *Journal of Chemical Information and Modeling*, *38*(15). doi: 10.1002/chin.200715216

Fawcett, T. (2006). An introduction to roc analysis. Pattern recognition letters, 27(8), 861–874.

Gardner, W., Lidz, C. W., Mulvey, E. P., & Shaw, E. C. (1996). A comparison of actuarial methods for identifying repetitively violent patients with mental illnesses. *Law and Human Behavior, 20*(1), 35.

Grove, W. M., & Meehl, P. E. (1996). Comparative efficiency of informal (subjective, impressionistic) and formal (mechanical, algorithmic) prediction procedures: The clinical-statistical controversy. *Psychology, Public Policy, and Law, 2*, 293–323.

Hare, R. (1991). *Manual for the Hare Psychopathy Checklist-Revised (PCL-R)*. 10.1037/t04993-000.

Helmus, L. M. & Babchishin, K. M. (2017). Primer on risk assessment and the statistics used to evaluate its accuracy. *Criminal Justice and Behavior, 44*(1), 8–25.

Howard, P. D. (2017). The effect of sample heterogeneity and risk categorization on area under the curve predictive validity metrics. *Criminal Justice and Behavior, 44*(1),103–120.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning with applications in r*. NY: Springer.

Kuhn, M. (2019, March 27). The caret Package. Retrieved February 6, 2020, from https://topepo.github.io/caret/subsampling-for-class-imbalances.html

Kuhn, M. & Johnson, K. (2013). *Applied predictive modeling.* NY: Springer.

LeDell, E., Gill, N., Aiello, S., Fu, A., Candel, A., Click, C., Kraljevic, T., Nykodym, T., Aboyoun, P., Kurka, M., & Malohlava, M. (2019). h2o: R Interface for 'H2O'. R package version 3.26.0.2. https://CRAN.R-project.org/package=h2o

Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G., & Ziegler, A. (2012). Probability Machines. *Methods of Information in Medicine*, *51*(01), 74–81. doi: 10.3414/me00-01-0052

McInnes, K. (2019, December 6). State and Capital Punishment. Retrieved February 1, 2020, from https://www.ncsl.org/research/civil-and-criminal-justice/death-penalty.aspx

Meehl, P. (1954). *Clinical versus statistical prediction: A theoretical analysis and review of the evidence*. Princeton: Educational Testing Service.

Millon, T., Simonsen, E., Birket-Smith, M., & Davis, R. D. (Eds.). (2003). *Psychopathy antisocial, criminal, and violent behavior*. New York: Guilford Press.

Monahan, J. & Steadman, H. J. (1994). Toward a rejuvenation of risk assessment research. In J. Monahan &H. J. Steadman (Eds.), *Violence and mental disorder: Developments in risk assessment* (pp. 1–17). Chicago,IL: University of Chicago Press.

Monahan, J., Steadman, H. J., Robbins, P. C., Appelbaum, P., Banks, S., Grisso, T,. … Silver, E. (2005). An actuarial model of violence risk assessment for persons with mental disorders. *Psychiatric services, 56*(7), 810–815.

Mossman, D. (2013). Evaluating risk assessments using receiver operating characteristic analysis: Rationale, advantages, insights, and limitations. *Behavioral Sciences & the Law, 31*(1), 23–39.

Ozkan, T. (2017). *Predicting recidivism through machine learning* (Doctoral dissertation).

Parent, G., Guay, J.-P., & Knight, R. A. (2012). Can we do better? the assessment of risk of recidivism by adult sex offenders. *Criminal justice and behavior, 39*(12), 1647–1667.

Polaschek, D. (2012). *An appraisal of the Risk-Need-Responsivity (RNR) model of offender rehabilitation and its application in correctional treatment*. Legal and Criminological Psychology. 17. 1 - 17. 10.1111/j.2044-8333.2011.02038.x.

Rosenfeld, B. & Lewis, C. (2005). Assessing violence risk in stalking cases: A regression tree approach. *Law and human behavior, 29*(3), 343.

Serin, R. C., & Amos, N. L. (1995). The role of psychopathy in the assessment of dangerousness. *International Journal of Law and Psychiatry*, 18(2), 231–238. https://doi.org/10.1016/0160-2527(95)00008-6

Silver, E. & Chow-Martin, L. (2002). A multiple models approach to assessing recidivism risk: Implications for judicial decision making. *Criminal justice and behavior, 29*(5), 538–568.

Singh, J. P. (2013). Predictive validity performance indicators in violence risk assessment: A methodological primer. *Behavioral Sciences & the Law, 31*(1), 8–22.

Singh, J. P., Grann, M., & Fazel, S. (2011). A comparative study of violence risk assessment tools: A systematic review and metaregression analysis of 68 studies involving 25,980 participants. *Clinical Psychology Review, 31*(3), 499–513.

Skeem, J. L., Mulvey, E. P., Appelbaum, P., Banks, S.,Grisso, T., Silver, E., & Robbins, P. C. (2004). Identifying subtypes of civil psychiatric patients at high risk for violence. *Criminal Justice and Behavior, 31*(4), 392–437.

Stalans, L. J., Yarnold, P. R., Seng, M., Olson, D. E., & Repp,M. (2004). Identifying three types of violent offenders and predicting violent recidivism while on probation: A classification tree analysis. *Law and human behavior, 28*(3), 253.

Steadman, H. J., Silver, E., Monahan, J., Appelbaum, P. S.,Clark Robbins, P., Mulvey, E. P., . . . Banks, S. (2000).A classification tree approach to the development of actuarial violence risk assessment tools. *Law and human behavior, 24*(1), 83.

Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010). Cost-sensitive learning methods for imbalanced data. *The 2010 International Joint Conference on Neural Networks (IJCNN)*. doi: 10.1109/ijcnn.2010.5596486

Thiele, C. (2019). *Cutpointr: Determine and Evaluate Optimal Cutpoints in Binary Classification Tasks*. R package version 1.0.0. http://CRAN.R-project.org/package=cutpointr

Tollenaar, N. & Van der Heijden, P. (2013). Which method predicts recidivism best?: A comparison of statistical, machine learning and data mining predictive models. *Journal of the Royal Statistical Society: Series A (Statistics in Society), 176*(2), 565–584.

US Census Bureau. (n.d.). Data Profiles. Retrieved from https://www.census.gov/acs/www/data/data-tables-and-tools/data-profiles/2016/

Wang, E. W. (1998). *The use of structural modeling techniques to predict violence potential in mentally ill male prisoners* (Doctoral dissertation, Texas A & M University-Commerce).

Widiger, T. A., & Trull, T. J. (1994). Personality disorders and violence. In J. Monahan & H. J. Steadman (Eds.), *The John D. and Catherine T. MacArthur Foundation series on mental health and development. Violence and mental disorder: Developments in risk assessment* (p. 203–226). University of Chicago Press.

Wintrup A. (1994). *The predictive validity of the PCL-R in high-risk mentally disordered offenders*. Unpublished manuscript, Simon Fraser University, Burnaby, BritishColumbia, Canada.

Wright, M. N., & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C and R. *Journal of Statistical Software*, *77*(1). doi: 10.18637/jss.v077.i01

Yang, M., Wong, S. C., & Coid, J. (2010). The efficacy of violence prediction: A meta-analytic comparison of nine risk assessment tools. *Psychological Bulletin, 136*(5), 740–767.

Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer, 3*(1), 32–35.

Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. *Third IEEE International Conference on Data Mining*. doi: 10.1109/icdm.2003.1250950

Zhou, Z.-H., & Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, *18*(1), 63–77. doi: 10.1109/tkde.2006.17

APPENDICES

A. CONFUSION MATRICES

C5.0 Down-Sampled

Table 11: C5.0 Down-Sampled Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,149	13,603
Predicted No	375	34,105

Table 12: C5.0 Down-Sampled 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,149	4,212
Predicted No	1,375	43,496

Table 13: C5.0 Down-Sampled 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,761	8,472
Predicted No	763	39,236

C5.0 SMOTE

Table 14: C5.0 SMOTE Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,193	15,121
Predicted No	331	32,587

Table 15: C5.0 SMOTE 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,056	3,641
Predicted No	1,468	44,067

Table 16: C5.0 SMOTE 10:1 Cut-Point

	Actual Yes	Actual No

Predicted Yes	1,623	7,407
Predicted No	901	40,301

C5.0 10:1 Cost-Sensitive Learning

Table 17: C5.0 10:1 Cost Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,116	6,552
Predicted No	1,408	41,156

Extreme Gradient Boosting No Rebalancing

Table 18: XGBoost Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,161	13,395
Predicted No	363	34,313

Table 19: XGBoost 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,080	3,582
Predicted No	1,444	44,126

Table 20: XGBoost 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,820	8,546
Predicted No	704	39,162

Extreme Gradient Boosting Down-Sampled

Table 21: XGBoost Down-Sampled Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,152	13,571
Predicted No	372	34,137

Table 22: XGBoost Down-Sampled 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	989	3,350
Predicted No	1,535	44,358

Table 23: XGBoost Down-Sampled 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,782	8,722
Predicted No	742	38,986

Extreme Gradient Boosting SMOTE

Table 24: XGBoost SMOTE Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,157	13,384
Predicted No	367	34,324

Table 25: XGBoost SMOTE 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,061	3,653
Predicted No	1,463	44,055

Table 26: XGBoost SMOTE 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,918	9,689
Predicted No	606	38,019

Random Forest No Rebalancing

Table 27: Random Forest Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,147	15,880
Predicted No	377	31,828

Table 28: Random Forest 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	851	2,931
Predicted No	1,673	44,777

Table 29: Random Forest 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,550	7,997
Predicted No	974	39,711

Random Forest Down-Sampled

Table 30: Random Forest Down-Sampled Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,061	14,147
Predicted No	463	33,561

Table 31: Random Forest Down-Sampled 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	780	2,588
Predicted No	1,744	45,120

Table 32: Random Forest Down-Sampled 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,658	8,381
Predicted No	866	39,327

Random Forest SMOTE

Table 33: Random Forest SMOTE Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,015	13,788
Predicted No	509	33,920

Table 34: Random Forest SMOTE 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	880	2,985
Predicted No	1,644	44,723

Table 35: Random Forest SMOTE 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,559	8,034
Predicted No	965	39,674

Random Forest 5:1 Cost-Sensitive Learning

Table 36: Random Forest 5:1 Cost Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,043	15,496
Predicted No	481	32,212

Table 37: Random Forest 5:1 Cost 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,524	47,708
Predicted No	0	0

Table 38: Random Forest 5:1 Cost 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,524	47,708
Predicted No	0	0

Random Forest 10:1 Cost-Sensitive Learning

Table 39: Random Forest 10:1 Cost Youden Cut-Point

	Actual Yes	Actual No
Predicted Yes	1,986	15,644
Predicted No	538	32,064

Table 40: Random Forest 10:1 Cost 5:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,524	47,708
Predicted No	0	0

Table 41: Random Forest 10:1 Cost 10:1 Cut-Point

	Actual Yes	Actual No
Predicted Yes	2,524	47,708
Predicted No	0	0

B. CODE

```
# Caret C5.0
# Sam Meeks
library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                            paste(x, collapse = " + "), sep = " ~"))

# C5.0 --------------------

set.seed(42)
system.time(
  vio.convic.c5.0 <- caret::train(vio.convic.form,
                                  data = train,
                                  method = "C5.0",
                                  tuneGrid = data.frame(trials = 100,
                                                        winnow = F,
                                                        model = "rules"),
                                  trControl = cv,
                                  metric = "ROC",
                                  maximize = T)
)

vio.convic.c5.0

# save --------------------

save.image("c5.environment.rdata")
```

```
# Caret C5.0 Down-Sample
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          sampling = "down",
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                            paste(x, collapse = " + "), sep = " ~"))

# C5.0 --------------------

set.seed(42)
system.time(
  vio.convic.c5.0.ds <- caret::train(vio.convic.form,
                                     data = train,
                                     method = "C5.0",
                                     tuneGrid = data.frame(trials = 100,
                                                           winnow = F,
                                                           model = "rules"),
                                     trControl = cv,
                                     metric = "ROC",
                                     maximize = T)
) #

vio.convic.c5.0.ds

# save --------------------

save.image("c5.ds.environment.rdata")
```

```
# Caret C5.0 Smote
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          sampling = "smote",
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                            paste(x, collapse = " + "), sep = " ~"))

# C5.0 --------------------

set.seed(42)
system.time(
  vio.convic.c5.0.smt <- caret::train(vio.convic.form,
                               data = train,
                               method = "C5.0",
                               tuneGrid = data.frame(trials = 100,
                                                     winnow = F,
                                                     model = "rules"),
                               trControl = cv,
                               metric = "ROC",
                               maximize = T)
) #

vio.convic.c5.0.smt

# save --------------------

save.image("c5.smt.environment.rdata")
```

```
# C5.0 Cost 5:1
# Sam Meeks

library(C50)
library(doMC)
registerDoMC(cores = 3)

# load data ----------

load("Meeks/fy1013.vio.combined.rdata")

# Set up cost ----------------------

#levels(train$violent.conviction)

cost.matrix <- matrix(c(
  NA, 5,
  1, NA), 2, 2, byrow=TRUE)

rownames(cost.matrix) <- colnames(cost.matrix) <- c("No", "Yes")

#cost.matrix

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                              paste(x, collapse = " + "), sep = " ~"))

# C5.0 --------------------

set.seed(42)
system.time(
  vio.convic.c5.cost.five <- C5.0(formula = vio.convic.form,
                          data = train,
                          trials = 100,
                          rules = TRUE,
                          control = C5.0Control(winnow = TRUE),
                          costs = cost.matrix)
)

vio.convic.c5.cost

# save --------------------

save.image("c5.cost.environment.five.rdata")
```

```
# C5.0 Cost 10:1
# Sam Meeks

library(C50)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cost ----------------------

#levels(train$violent.conviction)

cost.matrix <- matrix(c(
  NA, 10,
  1, NA), 2, 2, byrow=TRUE)

rownames(cost.matrix) <- colnames(cost.matrix) <- c("No", "Yes")

#cost.matrix

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                              paste(x, collapse = " + "), sep = " ~"))

# C5.0 --------------------

set.seed(42)
system.time(
  vio.convic.c5.cost <- C5.0(formula = vio.convic.form,
                         data = train,
                         trials = 100,
                         rules = TRUE,
                         control = C5.0Control(winnow = TRUE),
                         costs = cost.matrix)
)

#vio.convic.c5.cost

# save --------------------

save.image("c5.cost.environment.ten.rdata")
```

```
# Caret Ranger
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

ranger.grid <- expand.grid(mtry = 16:20,
                           splitrule = "gini",
                           min.node.size = 1)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                                    paste(x, collapse = " + "), sep = " ~"))

# ranger -------------------------

set.seed(42)
system.time(
  vio.convic.ranger <- caret::train(vio.convic.form,
                                    data = train,
                                    method = "ranger",
                                    tuneGrid = ranger.grid,
                                    trControl = cv,
                                    metric = "ROC",
                                    maximize = T,
                                    verbose = F)
)

# vio.convic.ranger

# save --------------------
```

```
save.image("ranger.environment.rdata")

# Caret Ranger DS
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)


# load data --------------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          sampling = "down",
                          summaryFunction = twoClassSummary)

ranger.grid <- expand.grid(mtry = 16:20,
                           splitrule = "gini",
                           min.node.size = 1)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                                    paste(x, collapse = " + "), sep = " ~"))

# ranger -------------------------

set.seed(42)
system.time(
  vio.convic.ranger.ds <- caret::train(vio.convic.form,
                                       data = train,
                                       method = "ranger",
                                       tuneGrid = ranger.grid,
                                       trControl = cv,
                                       metric = "ROC",
                                       maximize = T,
                                       verbose = T)
)
```

```
vio.convic.ranger.ds

# save --------------------

save.image("ranger.ds.environment.rdata")

# Caret Ranger Smote
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)


# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          sampling = "smote",
                          summaryFunction = twoClassSummary)

ranger.grid <- expand.grid(mtry = 16:20,
                           splitrule = "gini",
                           min.node.size = 1)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                                    paste(x, collapse = " + "), sep = " ~"))

# ranger -------------------------

set.seed(42)
system.time(
  vio.convic.ranger.smt <- caret::train(vio.convic.form,
                                        data = train,
                                        method = "ranger",
                                        tuneGrid = ranger.grid,
                                        trControl = cv,
```

```
                        metric = "ROC",
                        maximize = T,
                        verbose = T)
)

vio.convic.ranger.smt

# save --------------------

save.image("ranger.smt.environment.rdata")

# RF Cost 5:1
# Sam Meeks

library(randomForest)
library(doMC)
registerDoMC(cores = 36)


# load data ---------------

load("fy1013.vio.combined.rdata")

# Formula -------------------


x <- x[-10]

vio.convic.form <- as.formula(paste(y,
                        paste(x, collapse = " + "), sep = " ~"))



# random forest -------------------------

#levels(train$violent.conviction)


set.seed(42)
system.time(
  vio.convic.ranger.cost.five <- randomForest(formula = vio.convic.form,
                                  data = train,
                                  ntree=1000,
                                  classwt = c(1,5))
)
```

```
# vio.convic.ranger.cost

# save --------------------

save.image("ranger.cost.environment.five.rdata")

# RF Cost 10:1
# Sam Meeks

library(randomForest)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Formula ------------------


x <- x[-10]

vio.convic.form <- as.formula(paste(y,
                              paste(x, collapse = " + "), sep = " ~"))

# random forest -------------------------

#levels(train$violent.conviction)


set.seed(42)
system.time(
  vio.convic.ranger.cost <- randomForest(formula = vio.convic.form,
                                  data = train,
                                  ntree=1000,
                                  classwt = c(10,1))
)



# vio.convic.ranger.cost

# save --------------------

save.image("ranger.cost.environment.ten.rdata")
```

```
# Caret XGB
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                                    paste(x, collapse = " + "), sep = " ~"))

# XGB --------------------

nrounds <- 1000

xgb.tune <- expand.grid(
  nrounds = seq(from = 200, to = nrounds, by = 50),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(2, 3, 4, 5, 6),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

set.seed(42)
system.time(
  vio.convic.xgb <- caret::train(vio.convic.form,
                                 data = train,
                                 method = "xgbTree",
                                 tuneGrid = xgb.tune,
                                 trControl = cv,
                                 metric = "ROC",
```

```
                        maximize = T)
)

vio.convic.xgb

# save --------------------

save.image("xgb.environment.rdata")

# Caret XGB Down-Sample
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          sampling = "down",
                          summaryFunction = twoClassSummary)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                              paste(x, collapse = " + "), sep = " ~"))

# XGB --------------------

nrounds <- 1000

xgb.tune <- expand.grid(
  nrounds = seq(from = 200, to = nrounds, by = 50),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(2, 3, 4, 5, 6),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
```

```
)

set.seed(42)
system.time(
  vio.convic.xgb.ds <- caret::train(vio.convic.form,
                               data = train,
                               method = "xgbTree",
                               tuneGrid = xgb.tune,
                               trControl = cv,
                               metric = "ROC",
                               maximize = T)
)

vio.convic.xgb.ds

# save --------------------

save.image("xgb.ds.environment.rdata")

# Caret XGB Smote
# Sam Meeks

library(caret)
library(doMC)
registerDoMC(cores = 36)

# load data ----------

load("fy1013.vio.combined.rdata")

# Set up cv grids ----------------------

cv <- caret::trainControl(method = "cv",
                          number = 10,
                          allowParallel = TRUE,
                          classProbs = TRUE,
                          sampling = "smote",
                          summaryFunction = twoClassSummary)

# Formula ------------------

vio.convic.form <- as.formula(paste(y,
                               paste(x, collapse = " + "), sep = " ~"))

# XGB --------------------
```

```
nrounds <- 1000

xgb.tune <- expand.grid(
  nrounds = seq(from = 200, to = nrounds, by = 50),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(2, 3, 4, 5, 6),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)


set.seed(42)
system.time(
  vio.convic.xgb.smt <- caret::train(vio.convic.form,
                                     data = train,
                                     method = "xgbTree",
                                     tuneGrid = xgb.tune,
                                     trControl = cv,
                                     metric = "ROC",
                                     maximize = T)
)

vio.convic.xgb.smt

# save --------------------

save.image("xgb.smt.environment.rdata")

# HPCC Post-Processing
# Sam Meeks

library(wangr)
library(randomForest)
library(C50)
library(cutpointr)
library(tidyverse)
library(plotROC)

# read environments -------------------------

# C5
load("Meeks/hpcc results/c5.environment.rdata")

# C5 DS
```

```
load("Meeks/hpcc results/c5.ds.environment.rdata")

# C5 SMT
load("Meeks/hpcc results/c5.smt.environment.rdata")

# C5 Cost 5
load("Meeks/hpcc results/c5.cost.environment.five.rdata")

# C5 Cost 10
load("Meeks/hpcc results/c5.cost.environment.ten.rdata")
vio.convic.c5.cost.ten <- vio.convic.c5.cost
rm(vio.convic.c5.cost)

# XGB
load("Meeks/hpcc results/xgb.environment.rdata")

# XGB Fixed
#load("Meeks/hpcc results/xgb.fix.environment.rdata")

# XGB DS
load("Meeks/hpcc results/xgb.ds.environment.rdata")

# XGB SMT
load("Meeks/hpcc results/xgb.smt.environment.rdata")

# Ranger
load("Meeks/hpcc results/ranger.environment.rdata")

# Ranger DS
load("Meeks/hpcc results/ranger.ds.environment.rdata")

# Ranger SMT
load("Meeks/hpcc results/ranger.smt.environment.rdata")

# Ranger Cost 5
load("Meeks/hpcc results/ranger.cost.environment.five.rdata") # classwt = c(1,5)

# Ranger Cost 10
load("Meeks/hpcc results/ranger.cost.environment.ten.rdata") # classwt = c(1,10)
vio.convic.ranger.cost.ten <- vio.convic.ranger.cost
rm(vio.convic.ranger.cost)


rm(list=setdiff(ls(), c("vio.convic.c5.0", "vio.convic.c5.0.ds",
                        "vio.convic.c5.0.smt",
                        "vio.convic.c5.cost.ten", "vio.convic.c5.cost.five",
```

```
                "vio.convic.xgb", "vio.convic.xgb.ds", "vio.convic.xgb.smt",
                "vio.convic.ranger", "vio.convic.ranger.ds",
                "vio.convic.ranger.smt", "vio.convic.ranger.cost.ten",
                "vio.convic.ranger.cost.five",
                "test")))

# Predicted Probabilities -----------------------------

# C5
tmp.c5 <- predict(vio.convic.c5.0, test, type = "prob")
tmp.c5 <- tmp.c5 %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.c5 <- cbind(test, tmp.c5)

# C5 DS
tmp.c5.ds <- predict(vio.convic.c5.0.ds, test, type = "prob")
tmp.c5.ds <- tmp.c5.ds %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.c5.ds <- cbind(test, tmp.c5.ds)

# C5 SMT
tmp.c5.smt <- predict(vio.convic.c5.0.smt, test, type = "prob")
tmp.c5.smt <- tmp.c5.smt %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.c5.smt <- cbind(test, tmp.c5.smt)

# C5 Cost Sens 5:1
tmp.c5.five.cost <- predict(vio.convic.c5.cost.five, test, type = "class")
tmp.c5.five.cost <- ifelse(tmp.c5.five.cost == "Yes", 1, 0)
table(tmp.c5.five.cost)
test.c5.five.cost.sens <- cbind(test, tmp.c5.five.cost)

# C5 Cost Sens 10:1
tmp.c5.ten.cost <- predict(vio.convic.c5.cost.ten, test, type = "class")
tmp.c5.ten.cost <- ifelse(tmp.c5.ten.cost == "Yes", 1, 0)
table(tmp.c5.ten.cost)
test.c5.ten.cost.sens <- cbind(test, tmp.c5.ten.cost)

# XGB
tmp.xgb <- predict(vio.convic.xgb, test, type = "prob")
tmp.xgb <- tmp.xgb %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
```

```
test.xgb <- cbind(test, tmp.xgb)


# XGB DS
tmp.xgb.ds <- predict(vio.convic.xgb.ds, test, type = "prob")
tmp.xgb.ds <- tmp.xgb.ds %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.xgb.ds <- cbind(test, tmp.xgb.ds)

# XGB SMT
tmp.xgb.smt <- predict(vio.convic.xgb.smt, test, type = "prob")
tmp.xgb.smt <- tmp.xgb.smt %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.xgb.smt <- cbind(test, tmp.xgb.smt)

# Ranger
tmp.ranger <- predict(vio.convic.ranger, test, type = "prob")
tmp.ranger <- tmp.ranger %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.ranger <- cbind(test, tmp.ranger)

# Ranger DS
tmp.ranger.ds <- predict(vio.convic.ranger.ds, test, type = "prob")
tmp.ranger.ds <- tmp.ranger.ds %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.ranger.ds <- cbind(test, tmp.ranger.ds)

# Ranger SMT
tmp.ranger.smt <- predict(vio.convic.ranger.smt, test, type = "prob")
tmp.ranger.smt <- tmp.ranger.smt %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.ranger.smt <- cbind(test, tmp.ranger.smt)

# Ranger Cost Sens 5
tmp.ranger.cost.five <- predict(vio.convic.ranger.cost.five, test, type = "prob")
tmp.ranger.cost.five <- as.data.frame(tmp.ranger.cost.five)
tmp.ranger.cost.five <- tmp.ranger.cost.five %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.ranger.five.cost.sens <- cbind(test, tmp.ranger.cost.five)
```

```
# Ranger Cost Sens 10
tmp.ranger.cost.ten <- predict(vio.convic.ranger.cost.ten, test, type = "prob")
tmp.ranger.cost.ten <- as.data.frame(tmp.ranger.cost.ten)
tmp.ranger.cost.ten <- tmp.ranger.cost.ten %>% wang.naming() %>%
  dplyr::rename(predicted.prob = yes) %>%
  dplyr::select(predicted.prob)
test.ranger.ten.cost.sens <- cbind(test, tmp.ranger.cost.ten)


rm(tmp.c5, tmp.c5.ds, tmp.c5.smt, tmp.c5.five.cost, tmp.c5.ten.cost,
   tmp.xgb, tmp.xgb.ds, tmp.xgb.smt,
   tmp.ranger, tmp.ranger.ds, tmp.ranger.smt, tmp.ranger.cost.ten, tmp.ranger.cost.five)


# Performance --------------------------------

# C5.0 #

# Youden cuts
test.c5.youden <- cutpointr::cutpointr(test.c5, x = predicted.prob, class =
violent.conviction,
                                       pos_class = "Yes", neg_class = "No",
                                       direction = ">=",
                                       method = maximize_metric,
                                       metric = youden)


test.c5.ds.youden <- cutpointr::cutpointr(test.c5.ds, x = predicted.prob, class =
violent.conviction,
                                          pos_class = "Yes", neg_class = "No",
                                          direction = ">=",
                                          method = maximize_metric,
                                          metric = youden)


test.c5.smt.youden <- cutpointr::cutpointr(test.c5.smt, x = predicted.prob, class =
violent.conviction,
                                           pos_class = "Yes", neg_class = "No",
                                           direction = ">=",
                                           method = maximize_metric,
                                           metric = youden)

test.c5.five.costsen.youden <- cutpointr::cutpointr(test.c5.five.cost.sens, x =
tmp.c5.five.cost, class = violent.conviction,
                                           pos_class = "Yes", neg_class = "No",
                                           direction = ">=",
```

```
                                    method = maximize_metric,
                                    metric = youden)

test.c5.ten.costsen.youden <- cutpointr::cutpointr(test.c5.ten.cost.sens, x =
tmp.c5.ten.cost, class = violent.conviction,
                                        pos_class = "Yes", neg_class = "No",
                                        direction = ">=",
                                        method = maximize_metric,
                                        metric = youden)

# Cost cuts
test.c5.costratio.five <- cutpointr::cutpointr(test.c5, x = predicted.prob, class =
violent.conviction,
                                     pos_class = "Yes", neg_class = "No",
                                     direction = ">=",
                                     method = minimize_metric,
                                     metric = misclassification_cost,
                                     cost_fp = 1,
                                     cost_fn = 5)

test.c5.costratio.ten <- cutpointr::cutpointr(test.c5, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = minimize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 10)


test.c5.ds.costratio.five <- cutpointr::cutpointr(test.c5.ds, x = predicted.prob, class =
violent.conviction,
                                     pos_class = "Yes", neg_class = "No",
                                     direction = ">=",
                                     method = minimize_metric,
                                     metric = misclassification_cost,
                                     cost_fp = 1,
                                     cost_fn = 5)

test.c5.ds.costratio.ten <- cutpointr::cutpointr(test.c5.ds, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = minimize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
```

```
                                cost_fn = 10)

test.c5.smt.costratio.five <- cutpointr::cutpointr(test.c5.smt, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = minimize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 5)

test.c5.smt.costratio.ten <- cutpointr::cutpointr(test.c5.smt, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = minimize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 10)

# XGB #

# Youden cuts
test.xgb.youden <- cutpointr::cutpointr(test.xgb, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = maximize_metric,
                                    metric = youden)


test.xgb.ds.youden <- cutpointr::cutpointr(test.xgb.ds, x = predicted.prob, class =
violent.conviction,
                                      pos_class = "Yes", neg_class = "No",
                                      direction = ">=",
                                      method = maximize_metric,
                                      metric = youden)


test.xgb.smt.youden <- cutpointr::cutpointr(test.xgb.smt, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = maximize_metric,
                                    metric = youden)
```

```
# Cost cuts
test.xgb.costratio.five <- cutpointr::cutpointr(test.xgb, x = predicted.prob, class =
violent.conviction,
                                pos_class = "Yes", neg_class = "No",
                                direction = ">=",
                                method = minimize_metric,
                                metric = misclassification_cost,
                                cost_fp = 1,
                                cost_fn = 5)

test.xgb.costratio.ten <- cutpointr::cutpointr(test.xgb, x = predicted.prob, class =
violent.conviction,
                                pos_class = "Yes", neg_class = "No",
                                direction = ">=",
                                method = minimize_metric,
                                metric = misclassification_cost,
                                cost_fp = 1,
                                cost_fn = 10)

test.xgb.ds.costratio.five <- cutpointr::cutpointr(test.xgb.ds, x = predicted.prob, class =
violent.conviction,
                                pos_class = "Yes", neg_class = "No",
                                direction = ">=",
                                method = minimize_metric,
                                metric = misclassification_cost,
                                cost_fp = 1,
                                cost_fn = 5)

test.xgb.ds.costratio.ten <- cutpointr::cutpointr(test.xgb.ds, x = predicted.prob, class =
violent.conviction,
                               pos_class = "Yes", neg_class = "No",
                               direction = ">=",
                               method = minimize_metric,
                               metric = misclassification_cost,
                               cost_fp = 1,
                               cost_fn = 10)

test.xgb.smt.costratio.five <- cutpointr::cutpointr(test.xgb.smt, x = predicted.prob, class =
violent.conviction,
                                  pos_class = "Yes", neg_class = "No",
                                  direction = ">=",
                                  method = minimize_metric,
                                  metric = misclassification_cost,
                                  cost_fp = 1,
                                  cost_fn = 5)
```

```
test.xgb.smt.costratio.ten <- cutpointr::cutpointr(test.xgb.smt, x = predicted.prob, class =
violent.conviction,
                                   pos_class = "Yes", neg_class = "No",
                                   direction = ">=",
                                   method = minimize_metric,
                                   metric = misclassification_cost,
                                   cost_fp = 1,
                                   cost_fn = 10)

# Ranger #

# Youden cutpoint
test.ranger.youden <- cutpointr::cutpointr(test.ranger, x = predicted.prob, class =
violent.conviction,
                                     pos_class = "Yes", neg_class = "No",
                                     direction = ">=",
                                     method = maximize_metric,
                                     metric = youden)

test.ranger.ds.youden <- cutpointr::cutpointr(test.ranger.ds, x = predicted.prob, class =
violent.conviction,
                                       pos_class = "Yes", neg_class = "No",
                                       direction = ">=",
                                       method = maximize_metric,
                                       metric = youden)


test.ranger.smt.youden <- cutpointr::cutpointr(test.ranger.smt, x = predicted.prob, class =
violent.conviction,
                                       pos_class = "Yes", neg_class = "No",
                                       direction = ">=",
                                       method = maximize_metric,
                                       metric = youden)

test.ranger.five.costsen.youden <- cutpointr::cutpointr(test.ranger.five.cost.sens, x =
predicted.prob, class = violent.conviction,
                                        pos_class = "Yes", neg_class = "No",
                                        direction = ">=",
                                        method = maximize_metric,
                                        metric = youden)

test.ranger.ten.costsen.youden <- cutpointr::cutpointr(test.ranger.ten.cost.sens, x =
predicted.prob, class = violent.conviction,
                                        pos_class = "Yes", neg_class = "No",
                                        direction = ">=",
                                        method = maximize_metric,
```

```
                                    metric = youden)


# Cost cuts
test.ranger.costratio.five <- cutpointr::cutpointr(test.ranger, x = predicted.prob, class =
violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = minimize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 5)

test.ranger.costratio.ten <- cutpointr::cutpointr(test.ranger, x = predicted.prob, class =
violent.conviction,
                                   pos_class = "Yes", neg_class = "No",
                                   direction = ">=",
                                   method = minimize_metric,
                                   metric = misclassification_cost,
                                   cost_fp = 1,
                                   cost_fn = 10)

test.ranger.ds.costratio.five <- cutpointr::cutpointr(test.ranger.ds, x = predicted.prob, class
= violent.conviction,
                                     pos_class = "Yes", neg_class = "No",
                                     direction = ">=",
                                     method = minimize_metric,
                                     metric = misclassification_cost,
                                     cost_fp = 1,
                                     cost_fn = 5)

test.ranger.ds.costratio.ten <- cutpointr::cutpointr(test.ranger.ds, x = predicted.prob, class
= violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = minimize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 10)

test.ranger.smt.costratio.five <- cutpointr::cutpointr(test.ranger.smt, x = predicted.prob,
class = violent.conviction,
                                      pos_class = "Yes", neg_class = "No",
                                      direction = ">=",
                                      method = minimize_metric,
                                      metric = misclassification_cost,
```

```
                                    cost_fp = 1,
                                    cost_fn = 5)

test.ranger.smt.costratio.ten <- cutpointr::cutpointr(test.ranger.smt, x = predicted.prob,
class = violent.conviction,
                                   pos_class = "Yes", neg_class = "No",
                                   direction = ">=",
                                   method = minimize_metric,
                                   metric = misclassification_cost,
                                   cost_fp = 1,
                                   cost_fn = 10)

test.ranger.five.costsen.five <- cutpointr::cutpointr(test.ranger.five.cost.sens, x =
predicted.prob, class = violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = maximize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 5)

test.ranger.five.costsen.ten <- cutpointr::cutpointr(test.ranger.five.cost.sens, x =
predicted.prob, class = violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = maximize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 10)

test.ranger.ten.costsen.five <- cutpointr::cutpointr(test.ranger.ten.cost.sens, x =
predicted.prob, class = violent.conviction,
                                    pos_class = "Yes", neg_class = "No",
                                    direction = ">=",
                                    method = maximize_metric,
                                    metric = misclassification_cost,
                                    cost_fp = 1,
                                    cost_fn = 5)

test.ranger.ten.costsen.ten <- cutpointr::cutpointr(test.ranger.ten.cost.sens, x =
predicted.prob, class = violent.conviction,
                                   pos_class = "Yes", neg_class = "No",
                                   direction = ">=",
                                   method = maximize_metric,
                                   metric = misclassification_cost,
                                   cost_fp = 1,
```

```
                                        cost_fn = 10)

save.image("Meeks/predict.environment.rdata")
load("Meeks/predict.environment.rdata")

# Outputs ------------------------------------

# C5 Youden
summary(test.c5.youden)

# C5 Cost 5:1
summary(test.c5.costratio.five)

# C5 Cost 10:1
summary(test.c5.costratio.ten)


# C5 DS Youden
summary(test.c5.ds.youden)

# C5 DS 5:1
summary(test.c5.ds.costratio.five)

# C5 DS 10:1
summary(test.c5.ds.costratio.ten)


# C5 SMT Youden
summary(test.c5.smt.youden)

# C5 SMT 5:1
summary(test.c5.smt.costratio.five)

# C5 SMT 10:1
summary(test.c5.smt.costratio.ten)


# C5 Cost Sens (5:1) Youden
summary(test.c5.five.costsen.youden)

# C5 Cost Sens (10:1) Youden
summary(test.c5.ten.costsen.youden)


# XGB Youden
summary(test.xgb.youden)
```

```
# XGB Cost 5:1
summary(test.xgb.costratio.five)

# XGB Cost 10:1
summary(test.xgb.costratio.ten)


# XGB DS Youden
summary(test.xgb.ds.youden)

# XGB DS Cost 5:1
summary(test.xgb.ds.costratio.five)

# XGB DS Cost 10:1
summary(test.xgb.ds.costratio.ten)


# XGB SMT Youden
summary(test.xgb.smt.youden)

# XGB SMT 5:1
summary(test.xgb.smt.costratio.five)

# XGB SMT 10:1
summary(test.xgb.smt.costratio.ten)


# Ranger Youden
summary(test.ranger.youden)

# Ranger Cost 5:1
summary(test.ranger.costratio.five)

# Ranger Cost 10:1
summary(test.ranger.costratio.ten)


# Ranger DS Youden
summary(test.ranger.ds.youden)

# Ranger DS Cost 5:1
summary(test.ranger.ds.costratio.five)

# Ranger DS Cost 10:1
summary(test.ranger.ds.costratio.ten)
```

```
# Ranger SMT Youden
summary(test.ranger.smt.youden)

# Ranger SMT 5:1
summary(test.ranger.smt.costratio.five)

# Ranger SMT 10:1
summary(test.ranger.smt.costratio.ten)


# random forest cost sens (10:1) Youden
summary(test.ranger.ten.costsen.youden)

# random forest cost sens (10:1) 5:1
summary(test.ranger.ten.costsen.five)

# random forest cost sens (10:1) 10:1
summary(test.ranger.ten.costsen.ten)

# random forest cost sens (5:1) Youden
summary(test.ranger.five.costsen.youden)

# random forest cost sens (5:1) 5:1
summary( test.ranger.five.costsen.five)

# random forest cost sens (5:1) 10:1
summary(test.ranger.five.costsen.ten)
```