An Analysis of Penalized Regression in High Dimensional Scenarios

Gabriel Ackall[1]*, Connor Shrader[2]*
Mentor: Dr. Seongtae Kim[3]

[1]Georgia Tech, Civil Engineering

[2]University of Central Florida, Mathematics

[3]NCA&T University, Mathematics and Statistics

*Authors contributed equally

October 8, 2021

Abstract

With the prevalence of big data in recent years, the importance of modeling high dimensional data and selecting influential features has increased greatly. High dimensional data is common in many fields such as genome decoding, rare disease identification, economic modeling, and environmental modeling. However, most traditional regression machine learning models are not designed to handle high dimensional data or conduct variable selection. In this paper, we investigate the use of penalized regression methods such as ridge, least absolute shrinkage and selection operation (lasso), elastic net (E-net), smoothly clipped absolute deviation (SCAD), and minimax concave penalty (MCP) compared to traditional machine learning models such as random forest, XGBoost, and support vector machines. We evaluate these models using factorial design methods for Monte Carlo simulations in 270 environments, with factors being the number of predictors, number of samples, signal to noise ratio, covariance matrix, and correlation strength. We also compare different models using empirical data to evaluate their viability in real-world scenarios. Since our models are regression models, we evaluate the models using the test mean squared error, variable selection accuracy, β-sensitivity, and β-specificity. From our investigation, our findings indicate that penalized regression models outperform more traditional machine learning algorithms in most high-dimensional situations or in situations with a low number of data observations. Machine learning models are not often compared to penalized regression methods and so our analysis helps to expand the scope of how penalized regression is used to help model data. Additionally, the analysis helps to create a greater understanding of the strengths and weaknesses of each model type and provide a reference for other researchers on which machine learning techniques they should use, depending on a range of factors and data environments.

Keywords: penalized regression, variable selection, classification, machine learning, large p small n problem, Monte Carlo simulations

Contents

1 Introduction

In the modern world, machine learning techniques such as random forest, gradient boosting, and support vector machines are often touted as versatile one-size-fits-all solutions when it comes to modeling big data [28]. This is due in part to tree based models such as XGBoost winning numerous machine learning competitions [28]. While this versatility is frequently the case, an increasingly common type of data set where there are more predictors than observations can pose challenges for these machine learning algorithms. In these situations, lesser known statistical modeling techniques that perform variable selection can potentially perform equivalently or even better than these machine learning techniques. However, there is a distinct lack of academia focusing on comparing these variable selection techniques with the more traditional machine learning techniques. This paper serves to help bridge that gap.

In these situations where there are more predictors, p, than observations, n, many traditional machine learning techniques either become infeasible to use or fail to give good predictions. The large number of predictors and small number of observations make it easy for such models to **overfit**, meaning that the models become fine tuned to the exact training data and instead of finding generalized patterns for a population of data, they find specific occurrences in the training data [22, 18]. Because of this, overfitted models are sensitive to new data which causes them to perform extremely well on the training data, but poorly on testing data or when deployed in the real world. Because a model's predictions in real world scenarios and on new data is the entire purpose of a model, it is very important to reduce overfitting so that predictive accuracy in these scenarios is maximized.

This paper investigates several methods to handle the large p, small n problem. First, We considered wrapper methods such as forward selection, backward selection, stepwise forward selection and stepwise backward selection using both Akaike information criterion (AIC) and Bayesian information criterion (BIC) as the stopping criteria for the models [1, 32]. These models fit several linear models using different subsets of predictors and selects the model that optimizes a specific metric. In addition, we studied penalized regression models such as ridge regression [21], least absolute shrinkage and selection operation (lasso) [34], elastic-net [42], smoothly clipped absolute deviation (SCAD) [16], and minimax concave penalty (MCP) [40]. These models simultaneously select important predictors and fit a linear model. Finally, we considered a few machine learning models: random forests (RF) [6], gradient boosting in the form of XGBoost [9], and support vector machine (SVM) models [10]. These models do not assume a linear relationship between a response and its predictors. To compare these different techniques, models were trained and evaluated using both Monte Carlo simulations and empirical genomic data.

Section 2 contains details about each model and details the implementation of these models for our study. Section 3 describes our simulation study design and results, while section 5 explains our empirical data analysis and results. Section 6 is a discussion of our results and Section 7 is the conclusion.

2 Methodology

2.1 Modeling Background

Suppose that we have p predictor variables $X_1, X_2, \ldots, X_p$ and one response variable Y that depends on some (or all) of the predictors. We assume that Y can be expressed as

$$Y = f(X_1, X_2, \ldots, X_p) + \epsilon \tag{1}$$

where f is a function and ϵ is an independent random error with mean zero. The goal of supervised modeling is to find a function $\hat{f}$ that is a suitable approximation for f using a **training set**. This study focuses on **regression modeling**, where the response Y is a number on a continuous interval.

In practice, the function f that relates the predictors to the response is complex. Most statistical models assume that f takes some particular form and estimates a function $\hat{f}$ of that form. For example, many regression models assume that f is a linear function of the predictors; that is, linear models assume that

$$f(X_1, X_2, \ldots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{2}$$

where $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are coefficients that the models attempt to estimate.

The most common method to estimate the coefficients in a linear model is with **ordinary least squares** (OLS), which selects the values $\beta_0, \beta_1, \ldots, \beta_p$ that minimize the residual sum of squares

$$\text{RSS} = \sum_{i=1}^{n} \left[y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip}) \right]^2 \tag{3}$$

OLS is common because it is the best linear unbiased estimator; that is, OLS has a lower variance than any other linear unbiased estimator [20, 18]. However, if the number of predictors p is large compared to the number of observations n, OLS will overfit to the training data. Furthermore, if p exceeds n, then the OLS has infinitely many solutions that simply interpolate the training data. In these cases, OLS becomes unreliable for making predictions on test data.

By using models that have a small amount of bias, the high variance of OLS can be mitigated. Liu et. al. in [26] describe three types of variable selection algorithms. **Filter methods** work by evaluating the ability for each individual predictors to predict the response; then, a model is fit using the predictors selected [30, 12]. **Wrapper methods** fit models using different subsets of predictors and choose the model that has the best performance. Finally, **embedded methods** perform variable selection during the model training process. This paper focuses on wrapper methods and embedded methods. In addition, we considered several non-linear machine learning methods to draw a comparison between linear regression models and machine learning models.

2.2 Subset Selection Methods

Subset selection methods are wrapper methods that attempt to find a subset of the predictors $X_1, X_2, \ldots, X_p$ that are most correlated with the response variable Y. These

algorithms usually fit models for many different subsets and choose the subset of predictors that results in the best model. Although subset selection techniques can be applied to many types of models, we will focus on subset selection with linear regression.

There are two main benefits to using subset selection methods. By reducing the set of available predictors to just those that are strongly related to the response, overfitting can be mitigated. Another benefit of subset selection is that it creates a more interpretable model. If a data set includes thousands of predictors but only a few are related to the response, a model found using subset selection will be easier to understand than a model that relies on all of the parameters.

Forward stepwise selection starts by fitting a model with none of the predictors (by simply estimating each observation to be the mean of the response). The algorithm then iteratively chooses the predictor that best increases the model fit until a stopping condition is met. **Backward stepwise selection** does the same thing, but starts with a full model and works backwards. In addition, **forward stepwise selection** and **backward stepwise selection** are hybrid techniques that can both add and remove predictors in each iteration. Note that backward selection and backward stepwise selection can only be used when $p < n$, since they rely on starting on a full OLS model.

2.3 Penalized Regression

In general, **penalized regression** works by fitting a model that punishes large coefficient estimates. By forcing coefficient values to shrink, the resulting model will have relatively low variance. All of the models discussed here can be used when there are more predictors than observations. Most, but not all, of these methods can also perform variable selection.

Almost all of the penalized regression methods in this paper solve an optimization problem of the form

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \Big[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \Big]^2 + \lambda \sum_{j=1}^{p} P(\beta_j) \right\} \tag{4}$$

where the first summation is the usual residual sum of squares, $\lambda \geq 0$ is a hyperparameter that controls the strength of the penalty, and $P(\beta)$ is a penalty function applied to each of the coefficients. The penalty is not applied to the intercept β_0. In general, $P(\beta)$ is an even function that is non-decreasing as $|\beta|$ increases. If $\lambda = 0$, then we have the usual ordinary least squares estimator. As λ increases, a stronger penalty is applied which will decrease the coefficient values. As λ approaches ∞, the model becomes an empty model where only the intercept term is non-zero.

A similar way to express penalized regression is with the form

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \Big[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \Big]^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} P(\beta_j) \leq t \tag{5}$$

where t is a hyperparameter [22, 18]. In this form, penalized regression fits the ordinary least squares under the condition that the sum of the penalties of the coefficients is kept below some threshold. It can be shown that for every value of λ from Equation 4, there is a value

of t from Equation 5 that gives equivalent coefficient estimates [22]. Most algorithms to fit penalized regression models use the form given in Equation 4, but the form in Equation 5 may be considered more intuitive.

When fitting penalized regression models, the choice of λ is very important. If λ is too small, then models may be overfit just like ordinary least squares; on the other hand, if λ is large, then the resulting models are highly biased and the resulting models may be underfit. In practice, the best value of λ can be selected using **cross validation** and **grid search**. This process involves splitting the training set into k disjoint subsets of equal size called **folds**. Then, k models are fitted for different values of λ, where one fold is used as a testing set and the other $(k-1)$ folds are used for training. The value of λ that results in the lowest test error can then be selected.

Ridge regression is a penalized regression model that uses the penalty function $P(\beta) = \beta^2$ [21]. One advantage of ridge regression is that it can handle highly correlated data better than ordinary least squares. When predictors are correlated with each other, some algorithms have a hard time distinguishing which predictors are actually related to the response.

One drawback of ridge regression is that it cannot perform variable selection. It can shrink coefficients towards zero, but it cannot set coefficients to be exactly zero.

The **least absolute shrinkage and selection operation**, commonly referred to as **lasso**, is a shrinkage method with a very similar form to ridge regression [34, 22]. The penalty function for lasso is $P(\beta) = |\beta|$.

Unlike ridge regression, lasso regression can perform variable selection by setting coefficients to zero. This makes lasso regression favorable when most predictors are not related to the response variable. On the other hand, if the response truly does depend on all of the predictors, then lasso regression may incorrectly set some coefficients to zero. In addition, lasso regression does not have a closed-form solution, but computing the coefficients is still efficient.

Figure 1 provides a visual explanation for why ridge regression cannot perform variable selection while lasso regression can. Suppose that there are $p = 2$ predictors. The red circle in Figure 1a represents the circle $\beta_1^2 + \beta_2^2 \leq t$, which is the penalty boundary for ridge regression using the form given in Equation 5 when $t = 1$. The red square in Figure 1b represents the boundary $|\beta_1| + |\beta_2| \leq t$ for lasso regression. The point in the center of the blue ellipses represents the values of β_1 and β_2 that ordinary least squares would estimate. Because this point is not within either of the red regions, ridge regression and lasso regression will give different coefficient estimates. The blue ellipses around this point represent contour curves for the residual sum of squares, which is a quadratic function of β_1 and β_2. The first intersection of these ellipses with the red regions represents the coefficient values selected by ridge and lasso regression, since it represents the point within the red region with the lowest residual sum of squares. Because the ridge regression boundary is round, the intersection in Figure 1a occurs when β_1 and β_2 are both positive. For lasso regression in Figure 1b, the intersection occurs at a corner where $\beta_1 = 0$. Thus, lasso regression has set β_1 to zero, while β_1 is non-zero for ridge regression.

Elastic-net regression uses both the ridge penalty and the lasso penalty at the same

time [42]. Elastic-net solves the optimization problem

$$\hat{\beta}^{\text{E-net}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \right]^2 + \lambda_2 \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \right\} \tag{6}$$

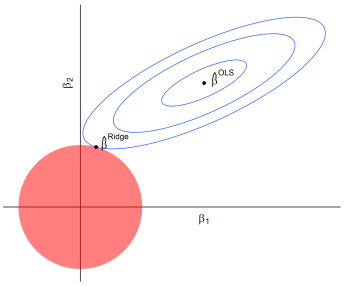
where λ_1 and λ_2 are both tuning parameters.

An important limitation to note is that elastic net performs best when it close to either ridge or lasso regression, meaning that either λ_1 greatly exceeds λ_2 or λ_2 greatly exceeds λ_1 [42]. Additionally, because elastic net requires two tuning parameters, this makes it much more difficult to determine the best combination of tuning parameters to minimize error in the regression. However, this problem has been largely solved through by the LARS-EN algorithm developed by Zou et. al. which efficiently solves for the tuning parameters [42].

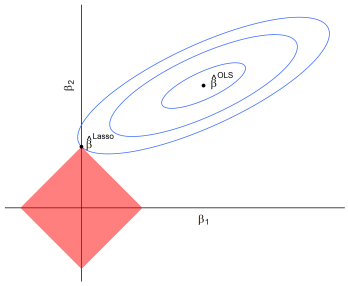
One major flaw of the lasso method is that the penalty punishes large coefficients, even if those coefficients should be large. One way to modify the lasso method is to use the **smoothly clipped absolute deviation** (SCAD) penalty [16]. The goal of this method is to punish large coefficients less severely, which can help mitigate some of the bias introduced by the lasso method. The penalty function $P(\beta)$ for SCAD satisfies

$$\frac{dP}{d\beta} = \text{sign}(\beta) \left[I(|\beta| < \lambda) + \frac{\max(a\lambda - |\beta|, 0)}{(a-1)\lambda} I(|\beta| > \lambda) \right] \tag{7}$$

where $a > 2$ is a new hyperparameter and I is the indicator function ($I(Q)$ equals 1 if a statement Q is true, and equals 0 if Q is false). If the hyperparameter a is large, then SCAD behaves like lasso for larger coefficient values. Also, notice that λ is an argument for the penalty function P.



(a) RSS contours and the ridge penalty boundary.



(b) RSS contours and the lasso penalty boundary.

Figure 1: RSS contours and penalty bounds for the ridge and lasso models when $p = 2$ and $t = 1$. The red regions represent the coefficient values allowed by ridge and lasso regression, respectively. The blue ellipses represent contours of the residual sum of squares, with $\hat{\beta}^{\text{OLS}}$ being the point where the residual sum of squares is minimized. The intersection of the ellipse with the red region in each plot represents the coefficient values selected by lasso and ridge.

An equivalent way to write the expression in Equation 7 is

$$\frac{dP}{d\beta} = \begin{cases} 1, & |\beta| \leq \lambda \\ \frac{a\lambda - |\beta|}{(a-1)\lambda}, & \lambda < |\beta| < a\lambda \\ 0, & \alpha\lambda < |\beta| \end{cases} \tag{8}$$

This penalty function does not punish coefficients with large magnitude as heavily as the lasso method. In fact, if the magnitude of a coefficient is larger than $a\lambda$, then the penalty becomes constant since the derivative becomes zero.

By integrating with respect to β and choosing $P(\beta) = 0$ [3], we see that

$$P(\beta) = \begin{cases} |\beta|, & |\beta| \leq \lambda \\ \frac{2a\lambda|\beta| - \beta^2 - \lambda^2}{2(a-1)\lambda}, & \lambda < |\beta| \leq a\lambda \\ \frac{\lambda(a+1)}{2}, & a\lambda < |\beta| \end{cases} \tag{9}$$

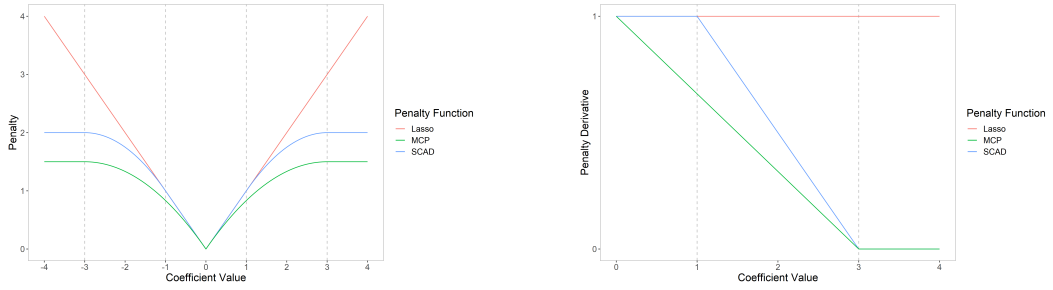
The **minimax concave penalty** (MCP) method is very similar to SCAD [40]. Both methods are used to avoid the high bias caused by the lasso method. MCP uses a penalty function that satisfies

$$\frac{dP}{d\beta} = \begin{cases} \text{sign}(\beta)\left(1 - \frac{|\beta|}{a\lambda}\right), & |\beta| \leq a\lambda \\ 0, & a\lambda < |\beta| \end{cases} \tag{10}$$

where, like SCAD, $a > 0$ is a hyperparameter. Integrating [3], we see that

$$P(\beta) = \begin{cases} |\beta| - \frac{\beta^2}{2a\lambda}, & |\beta| \leq a\lambda \\ \frac{1}{2}a\lambda, & a\lambda < |\beta| \end{cases} \tag{11}$$

Figure 2 below shows the penalty functions (and their derivatives) for lasso, SCAD, and MCP as a function of a coefficient value β. We see that lasso applies a much stronger penalty to large coefficients than SCAD or MCP. Also, note that SCAD starts with a derivative equal to that of the lasso for small values of β; on the other hand, the derivative of the penalty function for MCP starts decreasing immediately.



(a) Penalty functions for lasso, SCAD, and MCP.

(b) Derivatives of the penalty functions for lasso, SCAD, and MCP.

Figure 2: Penalty functions for lasso, SCAD, and MCP, as well as their derivatives. These plots use $\lambda = 1$ and $a = 3$. The dashed vertical lines are the knots for SCAD and MCP.

Note that of the penalized regression methods discussed above, only ridge regression has a general closed-form solution. Although lasso, elastic-net, SCAD and MCP do not have closed-form solutions, their solutions can be efficiently approximated [17, 4]. In some special cases, however, a closed-form solution exists.

Let $\mathbf{X}$ be a $n \times p$ matrix where each row contains the predictor values for one observation. To simplify things, we will assume that our data is centralized so that the coefficient β_0 is 0; that way, we do not need to include an extra entry of 1 in each row of $\mathbf{X}$. In an **orthonormal design**, we assume that $\mathbf{X}$ is orthonormal, meaning that the magnitude of each column of $\mathbf{X}$ is one and every pair of columns of $\mathbf{X}$ is orthogonal. In this special case, the matrix $(\mathbf{X}^\top \mathbf{X})^{-1}$ is the $p \times p$ identity matrix. As a result, each coefficient is independent of the other coefficients; in other words, we can compute the coefficient estimate for each predictor without needing to use the values for the other predictors. For example, in an orthonormal design, the general solution to ordinary least squares regression given in Equation **??** simplifies to

$$\hat{\boldsymbol{\beta}}^{\text{OLS}} = \mathbf{X}^\top \mathbf{y} \tag{12}$$

To compute the coefficient estimates for lasso, SCAD, and MCP in an orthonormal design, we first compute the vector $\mathbf{z} = \mathbf{X}^\top \mathbf{y}$ [16]. Each entry of $\mathbf{z}$ corresponds to one predictor. We then apply a **thresholding function** to each entry of $\mathbf{z}$ to get the coefficient estimate for that predictor. For lasso, elastic-net, SCAD, and MCP, a closed-form for this thresholding function exists [34, 16, 42, 40]. The input to the thresholding function is the actual coefficient value, and the output is the estimated value for that coefficient in an orthonormal design. Figure 3 shows the threshold functions for lasso, SCAD and MCP when $\lambda = 1$. For SCAD and MCP, we used the value $a = 3$. For reference, the identity line is included, which can be considered as the threshold function for ordinary least squares.

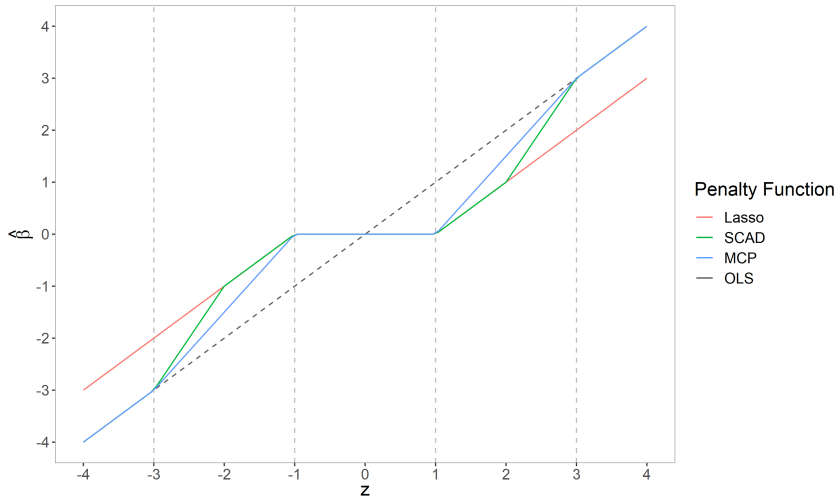


Figure 3: Thresholding function for lasso, SCAD, and MCP when $\lambda = 1$ and $a = 3$ in an orthonormal design. The input (z) represents an entry from the vector $\mathbf{z} = \mathbf{X}^\top \mathbf{y}$; the output ($\hat{\beta}$) is the coefficient estimate. The dashed vertical lines are the knots for SCAD and MCP.

We see that when $|z|$ is small, the thresholding function for lasso, SCAD, and MCP are

all zero. This matches with the assumption that these methods should set small coefficient estimates to zero. As $|z|$ increases, the estimated value for that coefficient gets larger. Notice that ordinary least squares doesn't decrease the estimated coefficient value for any choice of z; this is because of the fact that ordinary least squares is unbiased, whereas the other methods are biased.

In this figure, we also see that the threshold function for lasso is parallel to the identity line for large values of β. This means that even if a predictor is very influential on the response, lasso will predict a coefficient estimate that is less than the true coefficient value. This makes lasso a very biased model. This issue was one of the main motivations for SCAD and MCP. Unlike lasso, SCAD and MCP converge to the identity line as β increases, which makes these methods less likely to decrease large coefficient estimates.

Another feature of SCAD and MCP is their oracle-like properties [16, 40]. The term **oracle** in this context was first proposed by Donohe and Johnstone in [13]. Suppose that a linear model could be fitted with the aid of an oracle. This oracle could tell you the subset of predictors that are truly related to the response so that a model can be fitted using only the important predictors. Although such a model is not possible in practice, this oracle procedure serves an ideal that can be worked towards.

A linear model is said to have **oracle-like properties** if two asymptotic conditions hold [41]. Let p^* be the number of predictors that have non-zero coefficients. First, as $n \to \infty$, the model must be able to correctly identify the correct subset of non-zero predictors, which we will denote by $\mathcal{A}$. The second condition is that as $n \to \infty$, we must have

$$\sqrt{n}(\hat{\boldsymbol{\beta}}_{\mathcal{A}} - \boldsymbol{\beta}\mathcal{A}) \to_d \mathcal{N}_p^*(\mathbf{0}, \boldsymbol{\Sigma}^*) \tag{13}$$

where $\hat{\boldsymbol{\beta}}_{\mathcal{A}}$ are the coefficient estimates for the non-zero predictors, $\boldsymbol{\beta}_{\mathcal{A}}$ are the true coefficient values for the non-zero predictors, and $\mathcal{N}_p^*(\mathbf{0}, \boldsymbol{\Sigma}^*)$ is the multivariate normal distribution of the p^* non-zero predictors with mean zero and covariance matrix $\boldsymbol{\Sigma}^*$. In other words, as $n \to \infty$, the coefficient estimates for the non-zero predictors is normally distributed around the true coefficient values. Importantly, this means that the expected value for the coefficient estimates are the true coefficient values.

Both SCAD and MCP have oracle-like properties under certain conditions [16, 40]. Hence, SCAD and MCP can perform as well as the oracle estimator as n approaches ∞. On the other hand, lasso does not have oracle-like properties, meaning that its predictions cannot perform as well as a model found with the aid of an oracle.

2.4 Non-linear models

We next discuss several non-linear methods for regression: random forests, gradient boosting, and support vector machines.

Both random forest and gradient boosting models use **decision trees** to make predictions. A decision tree is a binary tree where each non-leaf node represents a condition and each leaf node represents a prediction value. To make a prediction, start at the root node and check whether the condition at that node is true or false. If true, move down to the node's first child; if false, move to the second child. This process is repeated until a leaf node is reached. This node will give a value that the decision tree predicts. Figure 4 gives an example of a decision tree with regression.

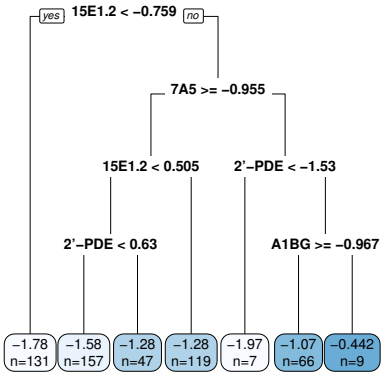


Figure 4: Example of a decision tree. This tree was generated using a `brTCGA` data set [39]. It attempts to predict the gene expression of the BRCA1 gene based on the expression of other genes. The decision tree was fitted and plotted with the `rpart` and `rpart.plot` libraries in `R`.

Although decision trees can be used as machine learning models on their own, it is more common to use decision trees in **ensemble methods**, which combine many different decision trees into a single model. This is because a single decision tree will usually have high variance since a small change in the training set can lead to a completely different decision tree [22].

Random Forests (RF) solve the issue of high variance by aggregating the predictions of many independent decision trees [6]. Each tree is fit using a subset of the observations and a subset of the predictors so that the trees are relatively independent from one another.

To fit each decision tree within a random forest model, we first select a random sample of the n observations with replacement, in a process called **bootstrapping** [15]. The number of observations chosen for each tree is a hyperparameter that can be changed. After selecting a set of observations, a random sample of predictors are chosen out of the p predictors without replacement. Again, this helps decorrelate the trees. The number of predictors used in each tree is also a hyperparameter that can be changed. A decision tree is then generated using the available observations and predictors.

A random forest model aggregates all of the individual decision trees into a single model. The number of trees generated, B, is a hyperparameter that can be changed; usually, a random forest model will contain at least several hundred trees. To make predictions with a random forest, a test observation is passed into each decision tree and the predictions from each tree are aggregated. Figure 5 demonstrates how a prediction is made using a random forest. For regression, the results are normally aggregated using the mean; for classification, the prediction chosen most often by the trees is usually used as the final prediction.

This idea of fitting multiple models with bootstrapping and aggregating the results can be used with other models besides decision trees. In general, this process is called **bagging** (a combination of the words "bootstrap" and "aggregating") [5].

Boosting is the technique of sequentially improving a weak learner until it becomes a

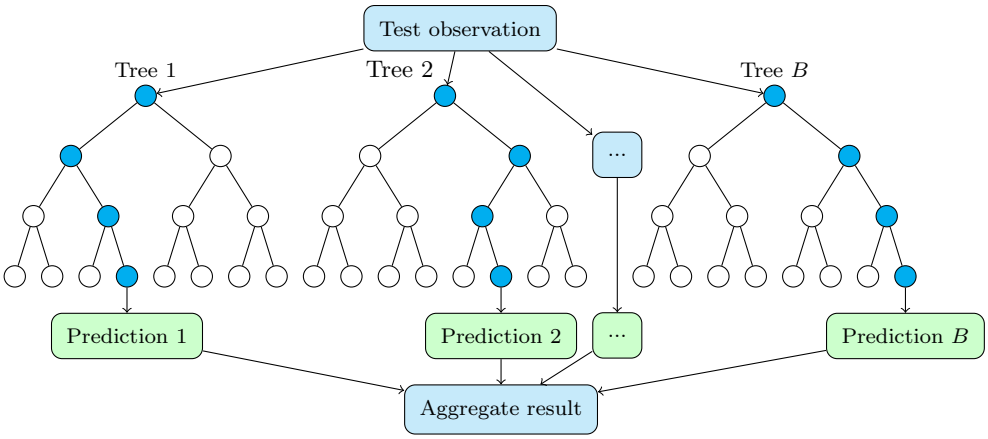


Figure 5: Visualization of how predictions are made using a random forest model. An observation is input into each decision tree. Predictions from each tree are then aggregated into a single result that is used as the final prediction.

strong learner [31]. A **gradient boosting machine** (GBM) is a boosting technique that uses gradient descent to minimize error in a model and correct the shortcomings of the previous model [19]. Boosting can be used on different types of models, but decision trees are the most common to use. Unlike random forest models, where each tree is independent of one another, the trees in a boosting model are grown sequentially. Each tree is fitted to correct the mistakes made by the previous tree. The details of how errors are corrected depends on the algorithm selected. With regression, each tree can be fitted by using the residuals from the previous tree as the training data [22]. Like random forest models, predictions are made by aggregating the results from each individual tree. See figure 6 for a diagram of how a boosting model is fitted.

Gradient boosting machines that use decision trees can be used for both regression

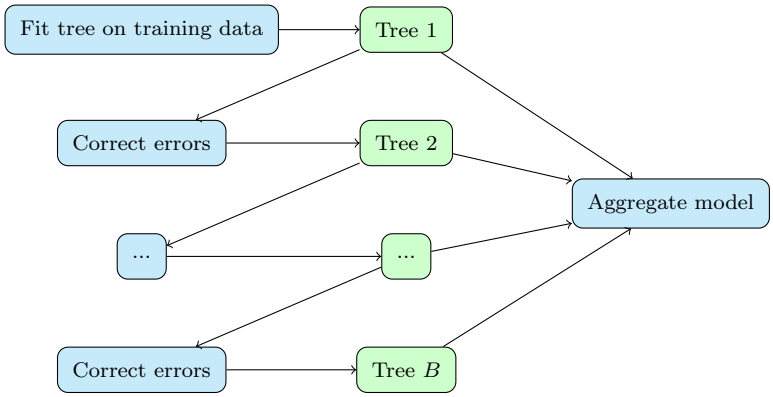


Figure 6: How a boosting model with decision trees is fitted. Each tree is fitted to correct the errors of the previous tree. Predictions are made by combining the results from each decision tree.

and classification. When using a gradient boosting model with decision trees, **relative variable importance** and **pruning** can be used as a sort of pseudo-variable selection method to lower complexity and prevent over-fitting. However, this does not yield the same interpretability or bias trade off benefits that true variable selection yields. When using a gradient boosting model with linear regression, there is the ability to use lasso, ridge, and elastic net penalized regression as the weak learner and perform variable selection, although this is much less commonly used than decision trees in gradient boosting.

Gradient boosting models often suffer from slow computation speeds due to the large number of sequential models that need to be trained. **Extreme Gradient Boosting** (XGBoost) is a faster version of gradient boosting that utilizes parallel computing as well as different optimization techniques to speed up computation [8]. For these reasons, XGBoost is often preferred over standard gradient boosting models and is very commonly used in many machine learning applications. In this study, we will use the XGBoost algorithm for gradient boosting.

There are a few different hyperparameters that can control how an XGBoost model is fit. The learning rate controls how quickly the gradient boosting model learns. If this learning rate is high, then the model learns quickly, but it may not learn as efficiently. If the learning rate is low, then the model takes longer but typically makes better predictions. The maximum tree depth determines how high each tree can be. Using smaller tree sizes may oversimplify the model, but it could also mitigate overfitting.

The final non-linear model that we considered is the **support vector machine** (SVM) [10]. Support vector machines are versatile models that can be used for both classification and regression. Originally, a support vector machine was designed as a binary classifier that separates the two response classes with a hyperplane in $(p-1)$-dimensional space. The hyperplane chosen by a support vector machine is chosen to maximize the distance between the hyperplane and any of the observation points. The observations closest to this separating hyperplane are called **support vectors**. Predictions are made by determining which side of the hyperplane an observation lies on. Observations that lie above the hyperplane are assigned to one class, and observations on the other side are predicted to be the other class. Figure 7 demonstrates a simple support vector machine for classification.

Note that in most data sets, the classes cannot be split perfectly into two sides. Furthermore, this model has very high variance as changing the points near the hyperplane can significantly alter the hyperplane. Finally, this model cannot handle cases where the true boundary is non-linear. Luckily, support vector machines in practice can handle such issues. Typically, support vector machines are allowed to misclassify some of the training data, which address the cases where the data cannot be split perfectly by a hyperplane. Also, the predictor space can be enlarged to handle non-linear decision boundaries; this is typically done by using **kernels**. For example, using a radial kernel can create decision boundaries that enclose regions of the p-dimensional space.

Support vector machines can also be generalized to handle classification when there are more than two classes. They can also be used for regression [14]. We will be using support vector machines for regression in our study.

Like the other non-linear methods, there are many different hyperparameters that can be tuned. The two hyperparameters that we considered are ϵ and the cost function. The hyperparameter ϵ defines how tolerant the algorithm is of small errors. If ϵ is large, then

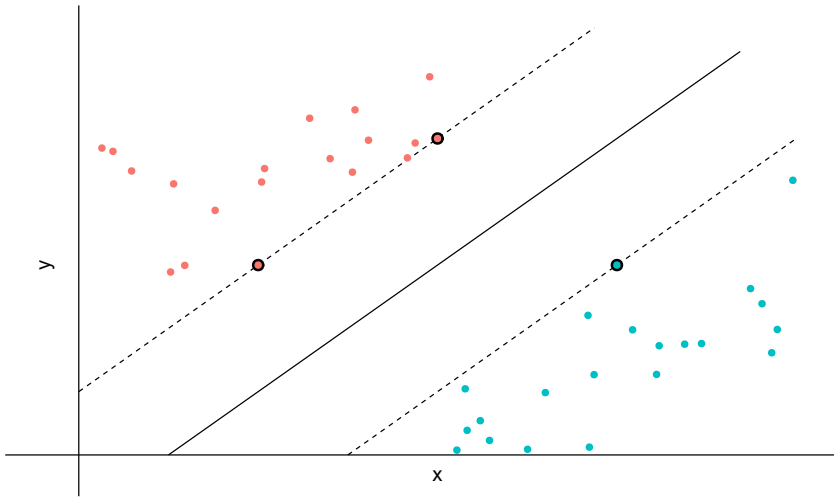


Figure 7: Visualization of a simple support vector machine for binary classification. The solid black line is the hyperplane that maximizes the margin between the two classes. The points with a black outline are the support vectors. They are the points that define the hyperplane.

the support vector machine will tolerate larger errors; if ϵ is small, then even small errors will be punished. The cost hyperparameter C determines how strongly the model punishes incorrect predictions. If C is large, then the model punishes incorrect predictions more, making it fit more tightly to the training set. If C is smaller, then the model is likely to allow more incorrect predictions in the training data.

2.5 Implementation

This section gives the specific details how we fit each model for both the simulated data and the empirical data.

All of our simulations were run using version 4.1.0 of `R`. Several different libraries were used to fit machine learning models using our simulated data. Table 1 summarizes the libraries used to fit models.

Ordinary least squares models were fitted using the `lm` function from the `stats` package in base `R`.

Subset selection models using forward, backward, stepwise forward, and stepwise backward selections were fitted using the `MASS` library. For each of these four algorithms, we fit models using both AIC and BIC, totaling to eight models.

For ridge, lasso, and elastic-net regression using `glmnet`, we used the `cv.glmnet` function. This function uses cross validation grid search to determine the value of λ that minimizes the cross validation error. Using cross validation can help generate a model that performs well on both training and testing data. We used the default value of 10 folds to fit models using `glmnet`. For elastic-net regression, we used the hyperparameter $\alpha = 0.8$ in our simu-

Table 1: `R` Libraries used and the models used from each library

Library	Models used	Version
`stats` [29]	Ordinary least squares	4.1.0
`MASS` [36]	Forward and backward selection	7.3-54
`glmnet` [17]	Ridge, lasso, elastic-net	4.1-1
`ncvreg` [4]	SCAD and MCP	3.13.0
`xgboost` [9]	Gradient boosting	1.4.1.1
`ranger` [37]	Random forest (simulations)	0.12.1
`randomForest` [25]	Random forest (empirical data)	4.6-14
`e1071` [27]	Support vector machine	1.7-7

lation study. This means that the elastic-net model is more similar to lasso (where $\alpha = 1$) than ridge (where $\alpha = 0$). This value of α was chosen so that the elastic-net model could emphasize the variable selection provided by lasso while also allowing it to handle multicollinearity from ridge. For the empirical data analysis, we used $\alpha = 0.5$. The remaining hyperparameters were given their default values.

For SCAD and MCP models, we used the `cv.ncvreg` function from the `ncvreg` library. Both SCAD and MCP depend on an additional hyperparameter a. We used the default values of a for both models: 3 for MCP and 3.7 for SCAD (note that the `ncvreg` documentation calls this hyperparameter γ instead of a). All other arguments were given their default values.

For the three non-linear models (gradient boosting, random forests, and support vector machines), we used cross validation and grid search to find suitable hyperparameters, and then fit a model using the full training set using the hyperparameters selected. Because many of the data sets used had large values of n and p, only the most important hyperparameters were tuned. This ensured that the models could be fit within a reasonable amount of time. All other hyperparameters were given their default values.

For gradient boosting with `xgboost`, we used different values for the learning rate (0.1, 0.3, and 0.5) and maximum tree depth (1, 3, and 7). A maximum of 1000 trees were generated, with an early stopping condition if the model failed to improve for 10 iterations in a row. We used five folds in the cross validation.

For random forests, we used `ranger` for the simulated data and `randomForest` for the empirical data. We did not use `ranger` for the empirical data because it could not handle the large number of predictors, resulting in stack overflow errors. For both `ranger` and `randomForest`, we tuned the number of predictors used per decision tree ($\lfloor\sqrt{p}\rfloor$, $\lfloor p/3\rfloor$, and $\lfloor p/2\rfloor$) and the number of trees (300, 400, 500 and 600). The best model was selected based on the out-of-bag error, which represents the average error for each observation using only the trees that did not include that observation.

Finally, for support vector machines using `e1071`, we varied ϵ (0.1, 0.5, 2), which affects the model's sensitivity to small errors. We also controlled the cost value C (0.5, 1, 2), which affects how much the model punishes wrong predictions.

Some of the models used could only be used for certain values of n and p. This is because either the runtime becomes unreasonable when n or p are large, or the model simply cannot

be used when p is too large. Ordinary least squares was used when $p \leq n$, since it cannot be used at all when $p > n$. For the same reason, the backward subset selection algorithms were also used when $p \leq n$. The forward subset selection algorithms were only used when $p \leq n$ and $p \leq 40$. When $p > 40$, the runtimes for forward selection and forward stepwise selection become unreasonably long due to problem of exponentially increasing combinations, so we chose to discard backward selection when p was too large. Ridge regression was used when $p \leq n$ as well as it is not designed for variable selection or for $p \geq n$ scenarios. Ridge was included in the empirical data study simply to further demonstrate its ineffectiveness in high dimensional settings.

Lasso, SCAD, MCP, GBM, and random forest models were used for all data sets. Support vector machine models were made for all of the simulated data but was not used for the empirical data. The empirical data has too many predictors which leads to unmitigable stack overflow errors. Simply put, the support vector machine model has trouble converging upon a local minimum and due to the need to store very large kernel matrices.

3 Monte Carlo Simulations

Monte Carlo simulations use randomly generated data to fit and test regression models. There are several benefits to using simulated data rather than experimental data. For one, the true relationship between the predictor variables and the response is known. Simulations can also be iterated many times, giving sturdier results about the effectiveness of each model. Finally, Monte Carlo simulations give us full control over how our data is distributed. This enables us to evaluate the models under various conditions.

3.1 Simulation Design

Our simulation study used two different functions for the response variable Y. Our first function assumed a linear relationship between the response and its predictors $X_1, X_2, \ldots, X_p$, while the second response used a non-linear relationship. By considering both linear and non-linear response functions, we obtain a more thorough understanding of how each model performs in different situations.

With our linear response function, we assumed that

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon \tag{14}$$

where β_0 is some intercept, $\beta_1, \ldots, \beta_p$ are coefficient values and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a normally distributed random error with mean 0 and variance σ^2. We chose the values $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = -2$, $\beta_5 = 0.5$ and $\beta_6 = 3$, while the remaining coefficient values were set to 0.

To generate the data, we first defined $\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots, \beta_p]^\top$, a $(p+1) \times 1$ vector of coefficient values. Next, we generated $\mathbf{X}$, a $n \times (p+1)$ matrix where each row contains the predictor values for one observation with an extra value of 1 in the first entry. The entries for the remaining columns of $\mathbf{X}$ were generated using the p-dimensional multivariate normal distribution $\mathcal{N}_p(0, \boldsymbol{\Sigma})$ with mean zero and covariance matrix $\boldsymbol{\Sigma}$. We assumed that every predictor had a standard deviation of 1, making the covariance matrix equivalent to

a correlation matrix. Four different correlation matrix structures were considered in our study, which are discussed later.

We then generated an $n \times 1$ error vector $\mathbf{e} \sim \mathcal{N}(0, \sigma^2)$ with mean zero and variance σ^2. The response $\mathbf{y}$ can then be computed by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e} \tag{15}$$

Given our chosen coefficient values, the response $\mathbf{y}$ becomes

$$\mathbf{y} = 1 + 2\mathbf{X}_1 - 2\mathbf{X}_2 + 0.5\mathbf{X}_5 + 3\mathbf{X}_6 + \mathbf{e} \tag{16}$$

where $\mathbf{X}_i$ is the i-th column of $\mathbf{X}$.

Our second response function calculated $\mathbf{y}$ as

$$\mathbf{y} = 6 \times 1_{\mathbf{X}_1 > 0} + \mathbf{X}_2^2 + 0.5\mathbf{X}_6 + 3\mathbf{X}_7 + 2 \times 1_{\mathbf{X}_8 > 0} \times 1_{\mathbf{X}_9 > 0} + \mathbf{e} \tag{17}$$

where $1_{\mathbf{X}_i > 0}$ is the index function defined by

$$1_{\mathbf{X}_i > 0} = \begin{cases} 0, & \mathbf{X}_i \leq 0 \\ 1, & \mathbf{X}_i > 0 \end{cases} \tag{18}$$

Note that this response function does have linear terms, but the function as a whole is non-linear.

Our simulation study used a **factorial design**. This means that we considered several factors that affect the simulated data in $\mathbf{X}$, each having multiple possible values. We then generated data using every possible combination of factor values, giving us a comprehensive assessment of model performance under various conditions. The factors that we controlled in our simulation are:

- The choice of response function (linear or non-linear)
- n, the number of observations (50, 200, and 1000),
- p, the number of predictors (10, 100, and 2000),
- σ, the standard deviation of the random error (1, 3, and 6),
- The correlation matrix structure (independent, symmetric compound, autoregressive, and blockwise), and
- ρ, the correlation between predictors (0.2, 0.5, and 0.9)

By taking every possible combination of these factors, we obtain $2 \times 3 \times 3 \times 3 \times 4 \times 3 = 648$ different settings for the simulations. However, because an independent correlation matrix does not have any correlation between predictors, the value of ρ is not used. Hence, we only needed to run 540 different settings. For each combination of factors, we ran 100 simulations. Each simulation randomly generated two data sets: one to train the various models, and one to test the models and evaluate performance. Both data sets contained n observations, meaning that a total of $2n$ observations were generated for each simulation.

As mentioned earlier, we considered four different covariance matrix structures. These structures determine the correlation between different predictors. If $\mathbf{\Sigma}$ is a correlation matrix, then $\mathbf{\Sigma}_{ij}$ (the entry at the i-th row and j-th column) represents the correlation between predictors i and j. If $\mathbf{\Sigma}_{ij} = 0$, there is no correlation; but if $\mathbf{\Sigma}_{ij} = 1$, then predictors i and j are perfectly correlated. Note that a correlation matrix is always symmetric, so $\mathbf{\Sigma}_{ij} = \mathbf{\Sigma}_{ji}$ for all indices i and j. This correlation can severely impact the performance of statistical models; if several predictors are highly correlated, then we expect the models to be less able to determine which predictors are actually related to the response.

The first correlation structure we considered is **independent correlation**. This means that the correlation matrix $\mathbf{\Sigma}$ has the form

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{19}$$

In other words, there is no correlation between different predictors, since $\mathbf{\Sigma}_{ij} = 0$ whenever $i \neq j$. Although this is a very simple case, it is very unrealistic; in the real world, predictors will typically be somewhat correlated with one another.

The next covariance structure is called **symmetric compound**. This structure has the form

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{bmatrix} \tag{20}$$

where $\rho \in [0, 1]$ is some correlation value. A symmetric compound covariance structure assumes that $\mathbf{\Sigma}_{ij} = \rho$ whenever $i \neq j$, meaning that all predictors are equally correlated with one another. By introducing correlation between different predictors, the data generated in our simulations is more realistic. However, a symmetric compound covariance matrix is still relatively simplistic. In real data sets, it is unrealistic to assume that all of the predictors have the exact same correlation with one another.

An autoregressive covariance structure assumes that

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & \rho & \cdots & \rho^{p-1} \\ \rho & 1 & \cdots & \rho^{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{p-1} & \rho^{p-2} & \cdots & 1 \end{bmatrix} \tag{21}$$

For any indices i and j, we have $\mathbf{\Sigma}_{ij} = \rho^{|i-j|}$. Consequently, each predictor is strongly correlated with nearby predictors and weakly correlated with more distant predictors. This form of covariance is commonly seen when using time series, since observed values at nearby times are likely to be highly correlated with one another.

Finally, a blockwise correlation matrix has the block-diagonal form

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{B}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{B}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{B}_k \end{bmatrix} \tag{22}$$

where 0 represents a block containing all zeroes, and each block $\mathbf{B}_i$ has a form identical to the symmetric compound matrix in Equation 20. This implies that predictors within the same block have correlation $\rho \in [0, 1]$, whereas predictors in different blocks have zero correlation. This type of correlation is more realistic than independent or symmetric compound correlation, since only certain groups of predictors are correlated with one another. One important consideration when using blockwise correlation is the size of each block. For our simulations, we used a block size of 5 when $p = 10$, a block size of 25 when $p = 100$, and a block size of 100 when $p = 2000$.

3.2 Evaluating Model Performance

We used four metrics to evaluate the performance of each model on the simulated data: **train mean squared error**, **test mean squared error**, β**-sensitivity** and β**-specificity**. The mean squared error (MSE) is computed by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{23}$$

where y_i is the value of the response and $\hat{y}_i$ is the predicted response value for observation i. In other words, the mean squared error is the average of the squared errors. The mean squared error was computed on both the n observations used to train the models and the n observations that were not used for training, giving us both a training error and a test error.

Because we are using simulated data, where the true response function is known, we can measure the β-sensitivity and β-specificity for each penalized linear regression model that performs variable selection [26]. For these models, we first measured the number of true positive, true negatives, false positives, and false negatives for the coefficient estimates. For any model, a coefficient estimate is a **true positive** if the coefficient is predicted to be non-zero when that predictor is actually related to the response value. The estimate is a **true negative** if the coefficient was correctly predicted to be zero when that predictor is not related to the response. A **false positive** happens when an important coefficient is incorrectly predicted to be non-zero. Finally, a coefficient estimate is a **false negative** if it was estimated to be zero but that predictor is actually related to the response. A model that perfectly identifies the important and unimportant predictors will have only true positives and true negatives.

For a particular model, let TP be the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives. Then the β-sensitivity model the model can be computed by

$$\beta\text{-sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{24}$$

The β-sensitivity is a measure of a model's ability to correctly identify predictors that are related to the response. If the β-sensitivity is close to 1, then the model assigns non-zero coefficients to all the important predictors; if instead the β-sensitivity is close to 0, then the model cannot identify important predictors well. Similarly, the β-specificity can be computed by

$$\beta\text{-specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{25}$$

The β-specificity measures a model's ability to recognize predictors that have no relationship with the response.

Because we ran simulations using 540 different combinations of factors, we will just highlight the results for the two extremes: when $n = 1000$ and $p = 10$, and when $n = 50$ and $p = 2000$. All of the plots in this section use facet plots to convey as much information as possible. Each plot measures the average value for one of the four metrics discussed above over 100 simulations. Each row represents a different value of σ, the standard deviation of the random error. Each column represents a correlation structure. The different shapes and colors for each point represent the strength of the correlation between predictors.

To save space in the plots, each model is given a shortened label. Many of these labels have been used throughout this paper; for example, ordinary least squares is labeled as OLS. The labels for the wrapper methods start with AIC or BIC, followed by either B for backward, SB for stepwise backward, F for forward, or SF for stepwise forward. As an example, the stepwise forward model evaluated with AIC is labeled as "AIC SF."

We begin by presenting the results from our simulations that used the linear response function, followed by the results from the non-linear response function.

4 Linear Simulation Results

Now, we will consider the opposite extreme where $n = 50$ and $p = 2000$. Figures **??** and **??** display the average mean squared error for simulations where $n = 50$ and $p = 2000$. Because less models were fitted for this combination of n and p, less models are shown in this figure. See Table 8 and Table 9 for a table of these results, including the standard deviations for each point.

We see that the mean squared error for lasso and elastic-net are generally larger than SCAD and MCP for both the training data and test data. Gradient boosting has almost zero training mean squared error under all conditions, but has a relatively large test error. Random forest and support vector machine models have a moderate training error but a large test error. Interestingly, we see that the non-linear models all perform better when there is a strong correlation between predictors. On the other hand, the linear models are somewhat less affected by the correlation.

Figure **??** shows the mean β-sensitivity for each model, with results shown in Table 10. We see that all of the models predict most of the non-zero coefficients when the correlation is low. When the correlation is high, all of the models struggle to identify the correct predictors. SCAD and MCP perform the best when the correlation is low but perform the worst when the correlation is high. Elastic-net performs particularly well compared to the other models when the correlation is high, especially when the correlation structure is

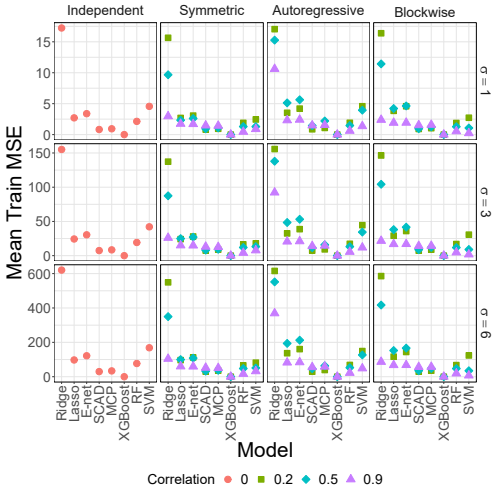


Figure 8: Average mean square error using training data for linear simulations when $n = 50$ and $p = 2000$.

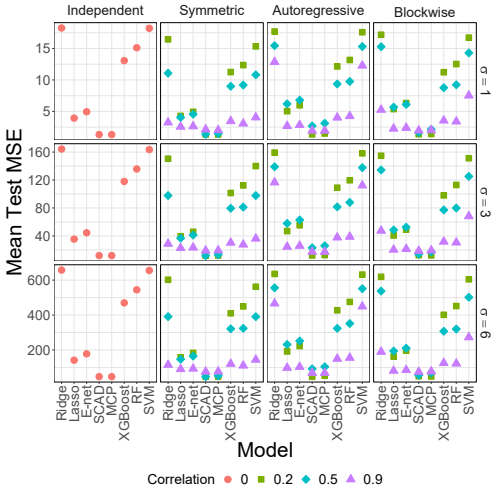


Figure 9: Average mean square error using testing data for linear simulations when $n = 50$ and $p = 2000$.

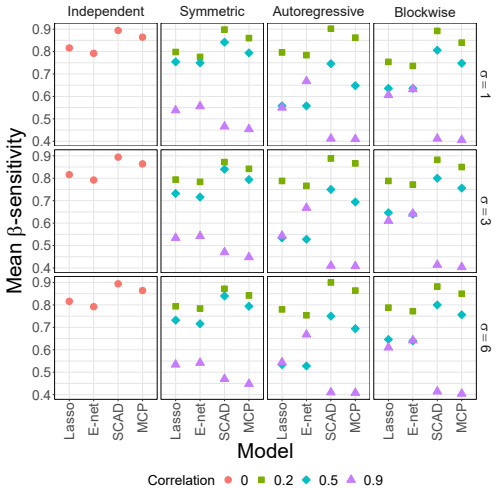


Figure 10: Average β-sensitivity for linear simulations when $n = 50$ and $p = 2000$.

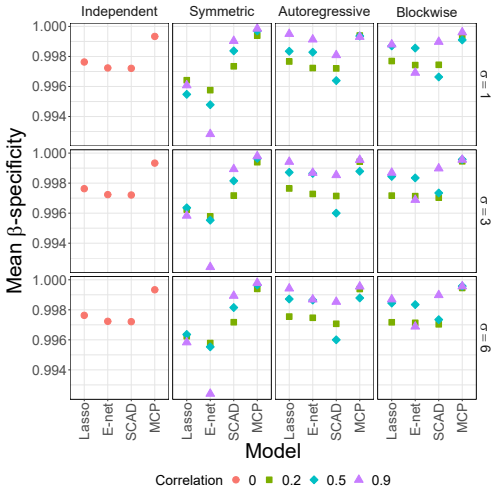


Figure 11: Average β-specificity for linear simulations when $n = 50$ and $p = 2000$.

autoregressive.

Figure **??** shows the average β-specificity $n = 50$ and $p = 2000$. See Table 11 for the corresponding results table. MCP appears to make the fewest mistakes when choosing zero coefficients. The performance of the other models depends heavily on the type of correlation and the correlation strength. Lasso and elastic-net perform the worst when the correlation structure is symmetric compound, whereas SCAD performs poorly when the correlation structure is autoregressive or blockwise. We also see that the models generally made less

mistakes as the correlation increases.

4.1 Non-linear Simulation Results

Now, we will highlight some results from the simulations that used the non-linear response function given by Equation 17.

Now, we will show the results for the non-linear simulations when $n = 50$ and $p = 2000$. Figures **??** and **??** show the average mean square errors on the training data and test data, respectively. We see that the linear and non-linear models have similar test mean squared errors. This differs from what we saw when $n = 1000$ and $p = 10$ (in Figure **??**) where the non-linear models significantly outperformed the linear models. Another interesting observation is that the non-linear models all have a noticeably lower test mean squared error when the correlation between predictors is high. The linear models still perform significantly worse on the training data compared to the test data.

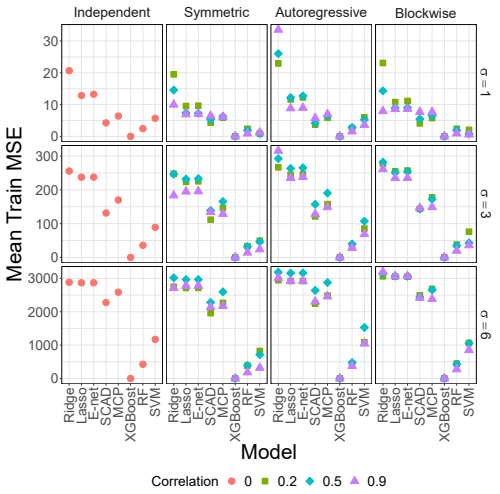


Figure 12: Average mean square error using training data for non-linear simulations when $n = 50$ and $p = 2000$.

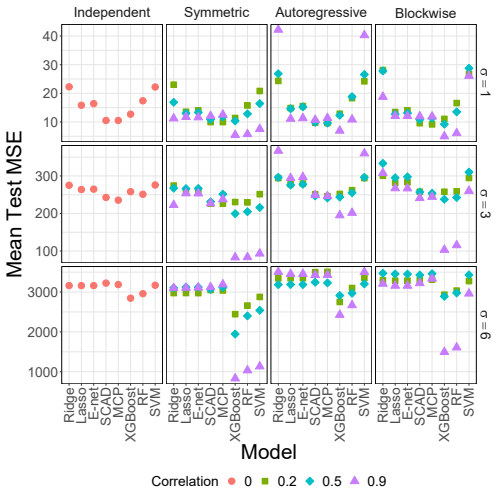


Figure 13: Average mean square error using testing data for non-linear simulations when $n = 50$ and $p = 2000$.

Figures **??** and **??** show the results for the β-sensitivity and β-specificity for the non-linear models when $n = 50$ and $p = 2000$. We see that all of the linear models estimate almost all of the coefficients as being equal to zero! SCAD and MCP were slightly more likely to correctly estimate non-zero coefficients as being non-zero, but they were also more likely to incorrectly identify unimportant predictors as having non-zero coefficients.

5 Empirical Data Analysis

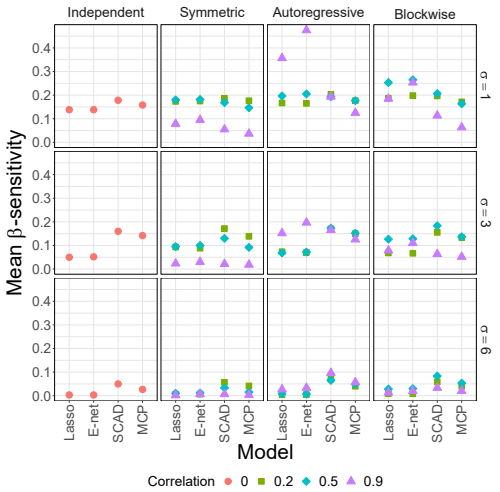


Figure 14: Average β-sensitivity for non-linear simulations when $n = 50$ and $p = 2000$.

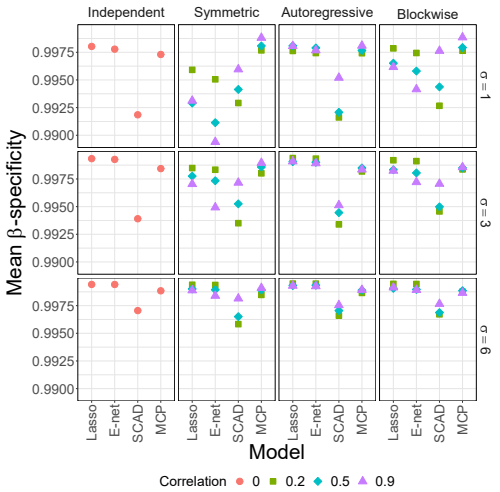


Figure 15: Average β-specificity for non-linear simulations when $n = 50$ and $p = 2000$.

5.1 Details of Empirical Data

For empirical data, we used the Breast Cancer database from The Cancer Genome Atlas (bcTCGA). A cleaned version of the data is provided by the `biglasso` `R` package [39]. This data set contains the gene expression data of 17323 genes from 536 patients. One of these genes is the BRCA1 gene which is among the first genes discovered that can increase the risk of breast cancer [24, 2]. Mutations in BRCA 1 and BRCA 2, another gene discovered 1 year after BRCA1, are responsible for two-thirds of breast cancer cases in women [11]. Because the BRCA1 gene interacts with other genes, it is useful to find genes that interact with BRCA1 to test in further studies [11]. The distribution of the BRCA1 gene expression levels in the bcTCGA database can be seen in Figure 16. The BRCA1 gene expression level will act as the output value in our regression analysis and the other 17322 genes will serve as predictor values.

This data is a prime example of the large p small n problem where there are many more predictors than data samples. Because of this, only penalized regression and machine learning techniques can be utilized. This is because there are more predictors than samples which makes least squares linear regression impossible and due to the high number of predictors, subset and stepwise regression becomes too computationally expensive to be feasible. Additionally, support vector machines struggle at such a high number of predictors and resulted in stack overflow errors which made fitting support vector machines on this data impossible.

To evaluate the models, we used **nested cross validation**. We first split the data into five folds. For each of these folds, we used the selected fold as a test set while the other four folds were used as a training set. We then fitted the models using cross validation on this training set, where one interior fold was used as a validation set while the other folds were used to train a model. The role of the validation set in the interior cross validation is different from the test set used in the exterior cross validation. In the interior cross

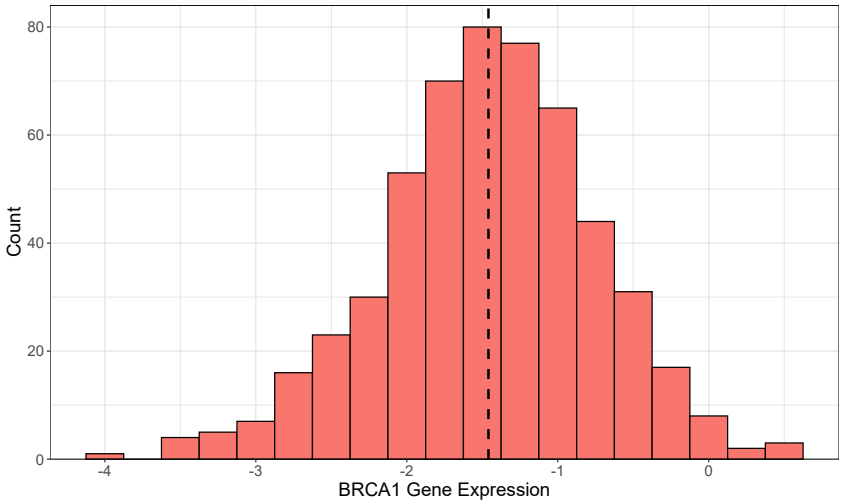


Figure 16: Distribution of BRCA1 gene expression levels. This is the variable that we used as the response. The mean gene expression is -1.459.

validation, the validation set is used to tune hyperparameters; the model that performs best against the validation set is then chosen. On the other hand, the test set in the exterior cross validation is not used to tune hyperparameters; its only purpose is to evaluate the performance of the models chosen in the inner cross validation. Because the outer test set is not used in the model fitting or selection process, it gives an unbiased evaluation of each model's performance.

We chose to use nested cross validation because it produces five models that were fitted using different subsets of the data for training and testing. If we had only fit one model, the subset of the data we choose for training and testing can have a huge impact on our findings. By using five models that are fit with different subsets of the data, we get a more accurate view of how each model performs in general. Cross validation also allows us to get an idea of how much variance each of these models has by comparing the results between different folds.

The hyperparameters tuned in each of the models were the same as those tuned in the Monte Carlo simulations. For ridge, lasso, elastic-net, SCAD, and MCP, we tuned the penalty strength λ; for elastic-net, we used the hyperparameter $\alpha = 0.5$, meaning that the penalty is in between that of lasso and ridge.

5.2 Empirical Data Results

Recall that we used nested cross validation when fitting models on the bcTCGA data set. This means that we fitted five models using different subsets of the data for training and testing. Figure 17 below shows a plot with the training and test mean squared error for every fold of every model. The bars show the average mean squared error for the five folds. In addition, Table 2 show the aggregated results for the train and test mean squared error.

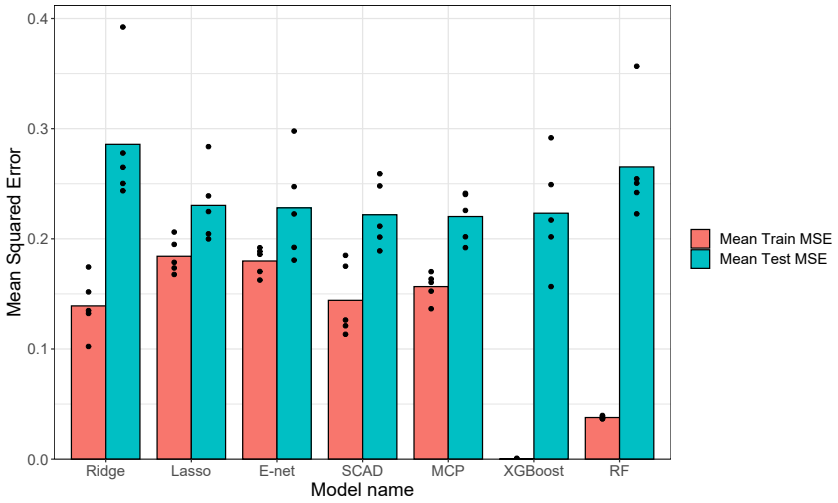


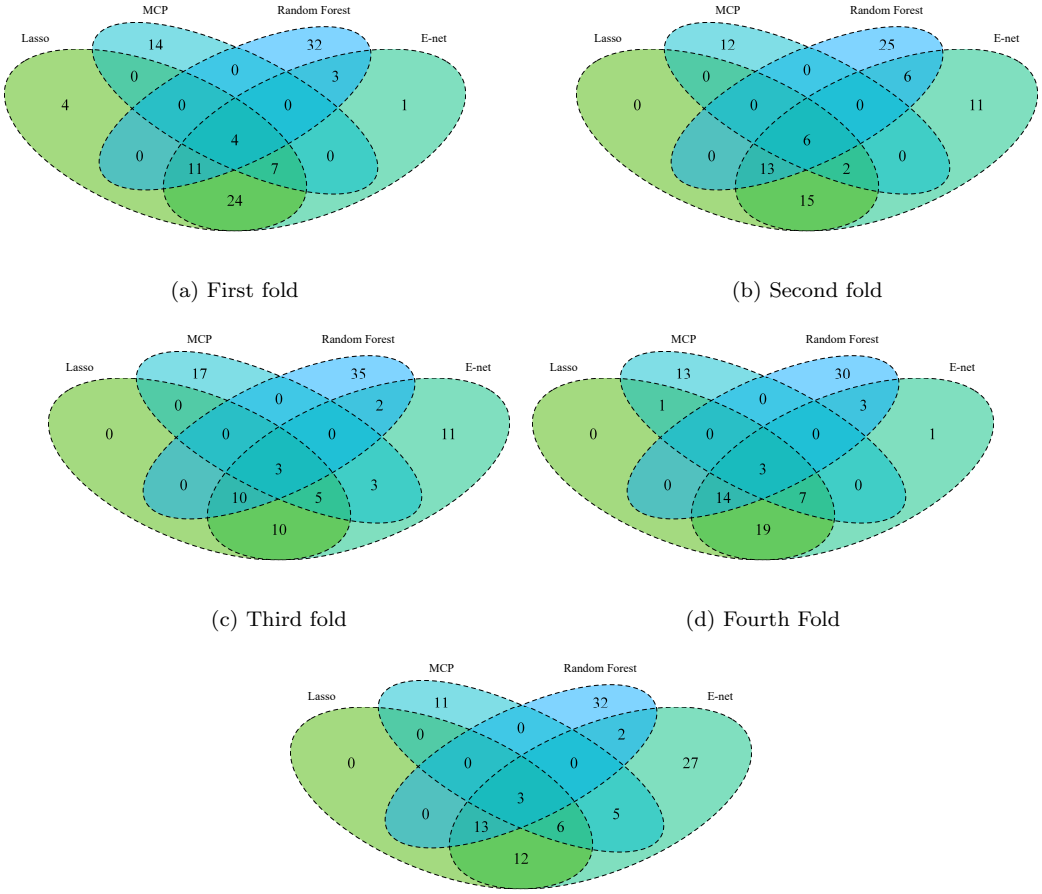
Figure 17: Mean squared error of the models fit on the bcTCGA data set. Each point represents the mean squared error for one fold, while the bars represent the average for the five folds.

Table 2: Average and standard deviation of the mean squared error of the models fit on the bcTCGA data set.

	Train MSE		**Test MSE**	
Model	**Average**	**SD**	**Average**	**SD**
Ridge	0.1391	0.0266	0.2858	0.0610
Lasso	0.1842	0.0159	0.2304	0.0337
E-net	0.1799	0.0127	0.2281	0.0469
SCAD	0.1442	0.0333	0.2218	0.0303
MCP	0.1566	0.0129	0.2202	0.0224
GBM	0.0002	0.0004	0.2233	0.0507
RF	0.0378	0.0013	0.2653	0.0525

The models can also be compared to each other by comparing the most important predictors selected by each model. Figure 18 shows five Venn diagrams. Each diagram shows the number of variables selected by lasso, elastic-net, MCP, and random forest as the most important for each of the folds. Figure 18a, for example, shows that four of the predictors were chosen by all four models as important. For lasso, elastic-net, and MCP, a predictor is considered important if its coefficient is non-zero. For random forest, we used the 50 predictors with the highest importance score (as computed by `randomForest`). We see that the number of predictors chosen by each model varies slightly among different folds, however, there still are common trends that are upheld between the different folds. The second fold in 18b had six predictors common to all four models, whereas the models tested in the third fold in 18c only had three.

Several of the genes in the data set appeared in all four models for several folds, indicating that they are very strongly related to the BRCA1 gene. The most important predictor was



(a) First fold

(b) Second fold

(c) Third fold

(d) Fourth Fold

(e) Fifth Fold

Figure 18: Venn diagrams of the predictors selected by lasso, elastic-net, MCP, and random forest models for each of the five folds. Each number represents the number of important predictors chosen by the models that overlap the number.

NBR2, which appeared in all four models for all five cross validation folds. This gene is a neighbor of the BRCA1 gene and is acts as a principal tumor suppressant [38]. Other significant genes that appeared in all four models for multiple folds include DTL (three folds), VPS25 (three folds), and CMTM5 (two folds). These three genes are also related to tumor suppression [23, 35, 33].

Table 3 shows the average runtime to fit each of the models on the bcTCGA data set.

Table 3: Average Runtimes for Empirical Data Models

Model	**Average Runtime** (seconds)
Ridge	29.29
Lasso	9.57
E-net	9.41
SCAD	17.28
MCP	15.15
GBM	538.12
RF	4906.59

6 Discussion

6.1 Discussion of Monte Carlo Results

Figures **??** and **??** shows that the best-performing models on test data were SCAD and MCP. The wrapper methods that used BIC performed better than SCAD and MCP when $n = 1000$ and $p = 10$, but could not be used in the high dimensional case where $n = 50$ and $p = 2000$. Lasso, elastic-net, and the wrapper methods with AIC all had low test mean squared errors as well. Ridge performed the worst out of all the linear models. The non-linear models also performed very badly on test data.

Notice that all the linear models except for ridge had a test mean squared error that was approximately equal to σ^2, the square of the random error. This indicates that these models had low bias and low variance, since most of their error was due to the random noise. On the other hand, ridge regression and the machine learning models had a test mean squared error that was much higher than σ^2, indicating that these models either had high bias or high variance.

Now, consider the training mean squared error results from Figures **??** and **??**. We see that training error for the linear models was roughly equal to the test error. This indicates that these models did not overfit to the training data. On the other hand, the non-linear models have far lower training mean squared errors than test errors. In particular, random forest models almost perfectly fit to the training data when $n = 1000$ and $p = 10$, while gradient boosting made an almost perfect fit when $n = 50$ and $p = 2000$.

Ridge regression, which performed badly on both training data and test data, was likely restricted by its penalty function and its inability to perform variable selection. Recall that three out of the five non-zero coefficient values chosen were greater than one. Because the penalty function for ridge squares each coefficient, ridge was very strongly discouraged from selecting coefficient estimates close to the true non-zero coefficient values. Furthermore, because ridge regression cannot set coefficients to zero and conduct variable selection, its coefficient estimates for the true non-zero coefficients were likely even smaller because unimportant predictors were given non-zero coefficient estimates. As a result, ridge regression was unable to predict the true coefficient values as well as the other linear models.

Another note about ridge regression is that it was more strongly affected by correlation between predictors when $n = 1000$ and $p = 10$, as seen in Figure **??**. The other linear models, which could set coefficients to zero, were likely able to eliminate unimportant predictors even

though they were correlated with the important predictors. Ridge regression, unable to eliminate these unimportant predictors, may have instead given larger coefficient estimates to the incorrect predictors.

Interestingly, the test mean squared error for random forest and support vector machine models improved as the correlation between predictors was increased. Figures **??** and **??** show that the test mean squared error for random forest and support vector machines decreased when correlation was increased. On the other hand, XGBoost was unaffected by an increase in the correlation value.

One possible explanation for this phenomenon is that the random forest models and support vector machines are less influenced by the random error when the correlation is large. For example, consider random forest models when $n = 50$ and $p = 2000$. When the correlation is zero, only a fraction of the decision trees generated in each model will contain any of the important predictors; many of the trees will instead be fit using only predictors that are unrelated to the response. This means that the random forest model is fitting with a large amount of noise. On the other hand, when the correlation is high, all of the trees will contain predictors that are correlated with the predictors with non-zero coefficients. As a result, none of the trees will truly be fitting on just random error. Similarly, for support vector machine models, the strong correlation may make it more difficult to fit using just the random noise.

Also note that changing the standard deviation of the random error seems to affect the mean squared errors of all the models equally. In addition, the correlation structure appears to have little influence on the training or test mean squared error.

Now, we will consider the β-sensitivity results for the linear models that could perform variable selection. These results are shown in Figures **??** and **??**. The β-sensitivity results when $n = 1000$ and $p = 10$ are rather uninteresting; all of the models correctly identified all of the non-zero coefficients except for a few cases where one coefficient was missed. The results for $n = 50$ and $p = 2000$ tell a richer story: SCAD and MCP tended to correctly identify more of the non-zero coefficients when the correlation between coefficients was low; however, when the correlation is high, SCAD and MCP found less than half of the non-zero coefficients, on average. Lasso and elastic-net performed slightly better when the correlation was high.

Looking at the β-specificity results from Figures **??** and **??**, we see that lasso, elastic-net, and the BIC wrapper methods had the highest β-specificity when $n = 1000$ and $p = 10$. When $n = 50$ and $p = 2000$, lasso and elastic-net had must lower β-specificity values compared to SCAD and MCP.

Another interesting result to note is that for both low-dimensional and high-dimensional settings, when the correlation between predictors increased, the β-specificity for SCAD and MCP typically increased. For elastic-net, a higher correlation caused the β-specificity to decrease when $n = 1000$ and $p = 10$; however, when $n = 50$ and $p = 2000$, the β-specificity of elastic-net depends heavily on the correlation structure. For a symmetric compound correlation structure, elastic-net had a much lower β-specificity as correlation increased. For an autoregressive or blockwise correlation structure, a higher correlation increased the β-specificity for elastic-net. This means that lasso and elastic-net have trouble identifying non-zero coefficients when all $p = 2000$ predictors are correlated to each other, while SCAD and MCP could handle this situation well.

Overall, we see that the ability for the penalized models to correctly identify whether coefficients are non-zero depends on how predictors are correlated with one another. However, the standard deviation of the random error does not appear to be too impactful on β-sensitivity or β-specificity.

Although wrapper methods and penalized regression models were the evident winners in this simulation study, it is worth noting that the non-linear models suffered from a few unfair disadvantages. For one, the simulated data assumed that there was a linear relationship between the predictors and response, which meant that the linear models were almost unbiased. The non-linear models were relatively more biased because they did not assume a linear relationship.

Another disadvantage for the non-linear models is that they can be controlled by many different hyperparameters, but we only considered a couple for each model. This was out of necessity; it would have been infeasible to fit models using more hyperparameters given the number of simulations that had to be run. If more hyperparameters could be included, then the performance of the non-linear models may have improved. At the same time, the more comprehensive tuning of several hyperparameters could act as a double edged sword in that it could increase the overfitting of these models. More hyperparameter tuning could lead a model to pick hyperparameters that fit the training data very precisely, but at the expense of increased testing error and computational cost.

6.2 Discussion of Empirical Data Results

In our empirical data study, SCAD and MCP also maintained the lowest testing mean squared error among the tested models. This can be seen in Figure 17 and Table 2. Furthermore, the mean squared error MCP had a relatively low standard deviation across the five folds. This may be due in part to the fact that MCP consistently selected the fewest number of predictors compared to Lasso and Elastic Net as shown by the Venn diagrams in Figure 18. Because this empirical data most likely contains multicollinearity, the fact that MCP selects fewer predictors may have helped to reduce overfitting. This is consistent with results from the Monte Carlo simulations which indicate that MCP performed better than its counterparts in multicollinear environments.

An interesting result was how few predictors were selected between all four models. It would be expected that the most important predictors would be chosen by all four models, but only three to six predictors were selected in common by all the models. Between MCP, Lasso, and Elastic Net, there were slightly more selected predictors in common, but these still were much less than expected. This is most likely because MCP is very different from the other regression techniques in that it uses non-convex optimization and thus selected many predictors that were different from all the other models. Since MCP had a much higher β-sensitivity and β-specificity rate compared to the other models as determined in the Monte Carlo simulations, it can be assumed that many of these predictors solely selected by MCP were significant predictors that were missed by the other models. This could have been another reason why MCP performed the best out of all the fitted models on the empirical data.

MCP also maintained the lowest testing MSE standard deviation for the empirical data as seen in Figure 2. This is to be expected given the bias/variance tradeoff. Because MCP

selects fewer predictors than the other models, it incurs greater bias but while also lessening variance. By definition, this means that MCP is less sensitive to changes in training data, such as changes in training data due to 5-fold cross validation.

The penalized regression models performed exceptionally faster than random forest and XGBoost as documented in Table 3. On average, Lasso and Elastic Net ran approximately 56x faster than XGBoost and 510x faster than random forest. MCP and SCAD ran approximately 30x faster than XGBoost and 290x faster than random forest. This provides a significant advantage to the penalized regression techniques, especially given that MCP and other penalized techniques performed better than random forest and XGBoost.

7 Conclusion

There is a severe lack of comprehensive testing comparing traditional machine learning methods such as random forest, gradient boosting, and support vector machines with penalized regression. Our paper bridges the divide between the machine learning and statistical fields in which these two types of models exist in. Testing using Monte Carlo simulations and empirical data has not been tested by other researchers with as many different environments and models.

These comparisons have shown that penalized regression should be added to the toolbox of any data scientist. In addition to performing better with less error than traditional machine learning techniques, penalized regression is much less computationally expensive. Additionally, in scenarios such as the empirical data study outlined earlier, penalized regression techniques can help determine the relationship between predictors and a response value. In such cases, the ability to determine these relationship can be more important than the predictive performance of a model.

However, there is still room for further comparison between these models. Our Monte Carlo simulation data was generated linearly, but we hope to also analyze non-linear Monte Carlo simulations and consider variable interaction. This would allow for comprehensive analysis of penalized regression even in unfavorable conditions that provide a clear advantage to non-linear machine learning techniques.

We could also run Monte Carlo simulations where the response is categorical rather than numerical. This could be used to study how penalized regression performs when used for classification data, which is the most common case for high dimensional data sets.

We are working on developing a hybrid technique between random forest and penalized regression. This method would harness the power of ensemble learning, while still being able to perform variable selection. This would hopefully perform better than either random forest or penalized regression methods individually. There are some models such as SREEM-forest [7] which conduct variable selection, however, these models use wrapper methods and variable importance which can be computationally expensive and do not inherently eliminate insignificant variables the same way penalized regression does. Given that random forest models are already very slow, performing additional stepwise selection may result in an exponentially slower runtime. Thus, it is important that inherent variable selection methods such as penalized regression methods are utilized in ensemble methods.

Lastly, we have developed a penalty function that has the potential to conduct variable selection while also handling multicollinearity well. The penalty function follows the L1/4 + L2 regularization and can be seen in Equation 26. We are interested in implementing this penalty function and analyzing its results. Given the power of penalized regression techniques seen in this study, it is important to develop new penalized regression techniques that improve on the drawbacks of previous designs.

$$\hat{\beta}^{\text{proposed}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \Big[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \Big]^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j|^{\frac{1}{4}} + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\} \tag{26}$$

8 Acknowledgments

This research was conducted as a part of the North Carolina A&T State University and Elon University Joint Summer REU Program in Mathematical Biology and was funded by the National Science Foundation DMS# 1851957/1851981.

The results here are in part based upon data generated by the TCGA Research Network: https://www.cancer.gov/tcga.

We would also like to thank Dr. Luke and Dr. Yokley and Yang Xue for their guidance throughout the research process.

References

[1] Akaike, H. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer, 1998.

[2] Antoniou, A., Pharoah, P. D., Narod, S., Risch, H. A., Eyfjord, J. E., Hopper, J. L., Loman, N., Olsson, H., Johannsson, O., Borg, Å., et al. Average risks of breast and ovarian cancer associated with brca1 or brca2 mutations detected in case series unselected for family history: a combined analysis of 22 studies. *The American Journal of Human Genetics*, 72(5):1117–1130, 2003.

[3] Breheny. Adaptive lasso, mcp, and scad. URL: https://myweb.uiowa.edu/pbreheny/7600/s16/notes/2-29.pdf, 2016.

[4] Breheny, P. and Huang, J. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, 5(1):232–253, 2011.

[5] Breiman, L. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Breiman, L. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Capitaine, L., Genuer, R., and Thiébaut, R. Random forests for high-dimensional longitudinal data. *Statistical Methods in Medical Research*, 30(1):166–184, 2021.

[8] Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[9] 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., and Li, Y. *xgboost: Extreme Gradient Boosting*, 2021. R package version 1.4.1.1.

[10] Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[11] Deng, C.-X. and Brodie, S. G. Roles of brca1 and its interacting proteins. *Bioessays*, 22(8):728–737, 2000.

[12] Ding, C. and Peng, H. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(02):185–205, 2005.

[13] Donoho, D. L. and Johnstone, J. M. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[14] Drucker, H., Burges, C. J., Kaufman, L., Smola, A., Vapnik, V., et al. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.

[15] Efron, B. and Tibshirani, R. J. *An introduction to the bootstrap*. CRC press, 1994.

[16] Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[17] Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[18] Friedman, J., Hastie, T., Tibshirani, R., et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[19] Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[20] Greene, W. H. *Econometric analysis*. Pearson Education India, 2003.

[21] Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[22] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An introduction to statistical learning*, volume 112. Springer, 2013.

[23] Kobayashi, H., Komatsu, S., Ichikawa, D., Kawaguchi, T., Hirajima, S., Miyamae, M., Okajima, W., Ohashi, T., Kosuga, T., Konishi, H., et al. Overexpression of denticleless e3 ubiquitin protein ligase homolog (dtl) is related to poor outcome in gastric carcinoma. *Oncotarget*, 6(34):36615, 2015.

[24] Kuchenbaecker, K. B., Hopper, J. L., Barnes, D. R., Phillips, K.-A., Mooij, T. M., Roos-Blom, M.-J., Jervis, S., Van Leeuwen, F. E., Milne, R. L., Andrieu, N., et al. Risks of breast, ovarian, and contralateral breast cancer for brca1 and brca2 mutation carriers. *Jama*, 317(23):2402–2416, 2017.

[25] Liaw, A. and Wiener, M. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[26] Liu, X.-Y., Wu, S.-B., Zeng, W.-Q., Yuan, Z.-J., and Xu, H.-B. Logsum+ l 2 penalized logistic regression model for biomarker selection and cancer classification. *Scientific Reports*, 10(1):1–16, 2020.

[27] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2021. R package version 1.7-7.

[28] Nielsen, D. Tree boosting with xgboost-why does xgboost win "every" machine learning competition? 2016.

[29] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[30] Sánchez-Maroño, N., Alonso-Betanzos, A., and Tombilla-Sanromán, M. Filter methods for feature selection–a comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 178–187. Springer, 2007.

[31] Schapire, R. E. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[32] Schwarz, G. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[33] Shao, L., Cui, Y., Li, H., Liu, Y., Zhao, H., Wang, Y., Zhang, Y., Ng, K. M., Han, W., Ma, D., et al. Cmtm5 exhibits tumor suppressor activities and is frequently silenced by methylation in carcinoma cell lines. *Clinical cancer research*, 13(19):5756–5762, 2007.

[34] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[35] Vaccari, T. and Bilder, D. The drosophila tumor suppressor vps25 prevents nonautonomous overproliferation by regulating notch trafficking. *Developmental cell*, 9(5):687–698, 2005.

[36] Venables, W. N. and Ripley, B. D. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[37] Wright, M. N. and Ziegler, A. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017.

[38] Xiao, Z.-D., Liu, X., Zhuang, L., and Gan, B. Nbr2: a former junk gene emerges as a key player in tumor suppression. *Molecular & cellular oncology*, 3(4):e1187322, 2016.

[39] Zeng, Y. and Breheny, P. The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in r. *ArXiv e-prints*, 2017.

[40] Zhang, C.-H. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[41] Zou, H. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

[42] Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

A Full Result Tables

The following pages contain the full table of Monte Carlo simulation results. Each table corresponds to one of the figures in Section **??**. The full results as well as relevant R code can be accessed through our Github repository.

Table 4: Mean and standard deviation of the mean squared error on training data when $n = 1000$ and $p = 10$. See Figure **??** for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	OLS	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04	0.99	0.04
	AIC B	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	BIC B	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	AIC SB	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	BIC SB	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	AIC F	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	BIC F	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	AIC SF	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	BIC SF	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	Ridge	1.11	0.05	1.13	0.05	1.19	0.05	1.41	0.05	1.12	0.05	1.18	0.05	1.37	0.05	1.12	0.05	1.18	0.05	1.39	0.05
	Lasso	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05
	E-net	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05	1.04	0.05
	SCAD	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	MCP	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04	1.00	0.04
	XGBoost	0.74	0.04	0.74	0.04	0.74	0.04	0.60	0.33	0.73	0.04	0.73	0.05	0.68	0.27	0.74	0.04	0.73	0.04	0.76	0.14
	RF	0.35	0.01	0.35	0.01	0.33	0.01	0.24	0.01	0.35	0.02	0.38	0.01	0.28	0.01	0.36	0.01	0.37	0.02	0.29	0.01
	SVM	0.45	0.03	0.50	0.05	0.69	0.11	0.91	0.05	0.47	0.06	0.57	0.08	0.86	0.06	0.48	0.03	0.63	0.10	0.86	0.06
3	OLS	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39	8.93	0.39
	AIC B	8.96	0.39	8.95	0.39	8.95	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39
	BIC B	8.99	0.40	8.99	0.39	8.99	0.39	8.98	0.39	8.98	0.39	8.99	0.39	8.98	0.39	8.99	0.39	8.99	0.39	8.99	0.39
	AIC SB	8.96	0.39	8.95	0.39	8.95	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39
	BIC SB	8.99	0.40	8.99	0.39	8.99	0.39	8.98	0.39	8.98	0.39	8.99	0.39	8.98	0.39	8.99	0.39	8.99	0.39	8.99	0.39
	AIC F	8.96	0.39	8.95	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39
	BIC F	8.99	0.40	8.99	0.39	8.99	0.39	8.98	0.39	8.98	0.39	8.99	0.39	8.99	0.39	8.99	0.39	8.99	0.39	8.99	0.39
	AIC SF	8.96	0.39	8.95	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39	8.96	0.39
	BIC SF	8.99	0.40	8.99	0.39	8.99	0.39	8.98	0.39	8.98	0.39	8.99	0.39	8.99	0.39	8.99	0.39	8.99	0.39	8.99	0.39
	Ridge	9.95	0.42	10.16	0.44	10.72	0.44	12.72	0.49	10.11	0.43	10.62	0.43	12.34	0.47	10.11	0.42	10.61	0.42	12.47	0.50
	Lasso	9.39	0.44	9.38	0.42	9.38	0.42	9.38	0.42	9.38	0.41	9.38	0.41	9.37	0.41	9.38	0.42	9.38	0.42	9.36	0.42
	E-net	9.37	0.43	9.37	0.43	9.36	0.43	9.36	0.43	9.36	0.43	9.36	0.43	9.35	0.43	9.37	0.43	9.36	0.43	9.34	0.43
	SCAD	8.97	0.39	8.97	0.40	8.97	0.39	8.97	0.39	8.97	0.39	8.97	0.40	8.97	0.39	8.97	0.40	8.97	0.40	8.97	0.39
	MCP	8.97	0.39	8.98	0.40	8.97	0.39	8.97	0.39	8.97	0.39	8.98	0.39	8.97	0.40	8.97	0.39	8.97	0.39	8.97	0.39
	XGBoost	6.61	0.34	6.61	0.34	6.63	0.38	5.62	2.82	6.62	0.33	6.61	0.38	6.28	2.09	6.65	0.34	6.57	0.41	6.31	2.09
	RF	3.14	0.13	3.20	0.14	2.99	0.14	2.14	0.09	3.19	0.12	3.38	0.13	2.51	0.11	3.21	0.13	3.39	0.14	2.64	0.12
	SVM	4.04	0.26	4.43	0.28	6.03	0.86	8.19	0.45	4.20	0.27	5.16	0.79	7.70	0.53	4.37	0.50	5.67	0.88	7.68	0.47
6	OLS	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56	35.73	1.56
	AIC B	35.83	1.56	35.82	1.56	35.82	1.56	35.84	1.57	35.83	1.57	35.82	1.56	35.82	1.56	35.83	1.56	35.82	1.56	35.83	1.56
	BIC B	35.95	1.60	35.95	1.57	35.94	1.57	35.93	1.58	35.93	1.57	35.95	1.57	35.93	1.57	35.94	1.58	35.95	1.57	35.95	1.57
	AIC SB	35.83	1.56	35.82	1.56	35.82	1.56	35.84	1.57	35.83	1.57	35.82	1.56	35.82	1.56	35.83	1.56	35.82	1.56	35.83	1.56
	BIC SB	35.95	1.60	35.95	1.57	35.94	1.57	35.93	1.58	35.93	1.57	35.95	1.57	35.93	1.57	35.94	1.58	35.95	1.57	35.95	1.57
	AIC F	35.83	1.56	35.82	1.56	35.83	1.56	35.84	1.57	35.83	1.56	35.83	1.56	35.85	1.56	35.83	1.56	35.83	1.57	35.84	1.56
	BIC F	35.95	1.60	35.95	1.57	35.94	1.57	35.93	1.58	35.93	1.57	35.95	1.56	35.94	1.58	35.94	1.58	35.95	1.57	35.95	1.57
	AIC SF	35.83	1.56	35.82	1.56	35.83	1.56	35.84	1.57	35.83	1.56	35.83	1.56	35.85	1.56	35.83	1.56	35.83	1.57	35.84	1.56
	BIC SF	35.95	1.60	35.95	1.57	35.94	1.57	35.93	1.58	35.93	1.57	35.95	1.56	35.94	1.58	35.94	1.58	35.95	1.57	35.95	1.57
	Ridge	39.78	1.68	40.64	1.76	42.89	1.78	50.87	1.96	40.45	1.73	42.46	1.73	49.34	1.89	40.43	1.70	42.46	1.68	49.89	2.00
	Lasso	37.54	1.68	37.51	1.67	37.51	1.66	37.51	1.67	37.52	1.66	37.50	1.66	37.46	1.66	37.52	1.66	37.50	1.67	37.44	1.67
	E-net	37.48	1.75	37.46	1.71	37.45	1.71	37.44	1.72	37.46	1.73	37.44	1.73	37.40	1.71	37.46	1.73	37.44	1.71	37.38	1.73
	SCAD	35.90	1.57	35.89	1.58	35.89	1.57	35.88	1.57	35.90	1.57	35.88	1.58	35.87	1.57	35.89	1.58	35.90	1.58	35.89	1.58
	MCP	35.89	1.57	35.90	1.58	35.89	1.58	35.90	1.58	35.89	1.57	35.91	1.58	35.88	1.58	35.89	1.57	35.90	1.57	35.88	1.56
	XGBoost	26.42	1.46	26.50	1.30	26.47	1.43	22.57	10.98	26.52	1.35	26.43	1.39	24.88	9.02	26.62	1.33	26.37	1.56	26.19	6.76
	RF	12.54	0.51	12.79	0.56	11.97	0.54	8.55	0.36	12.75	0.50	13.53	0.52	10.04	0.45	12.83	0.53	13.57	0.56	10.55	0.50
	SVM	16.16	1.04	17.72	1.11	24.16	3.62	32.84	1.76	16.81	1.08	20.60	3.25	30.84	2.09	17.26	1.03	22.55	3.45	30.80	1.94

Table 5: Mean and standard deviation of the mean squared error on test data when $n = 1000$ and $p = 10$. See Figure **??** for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	OLS	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	AIC B	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	BIC B	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	AIC SB	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	BIC SB	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	AIC F	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	BIC F	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	AIC SF	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	BIC SF	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	Ridge	1.13	0.06	1.15	0.06	1.21	0.06	1.43	0.07	1.14	0.06	1.20	0.07	1.40	0.06	1.14	0.06	1.20	0.06	1.40	0.08
	Lasso	1.06	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.06	0.05	1.05	0.05	1.05	0.05	1.05	0.05
	E-net	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05	1.05	0.05
	SCAD	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04
	MCP	1.01	0.05	1.01	0.04	1.01	0.05	1.01	0.05	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.04	1.01	0.05
	XGBoost	1.22	0.07	1.22	0.06	1.22	0.06	1.24	0.10	1.22	0.06	1.22	0.06	1.24	0.08	1.22	0.06	1.21	0.06	1.22	0.07
	RF	2.04	0.15	2.05	0.15	1.94	0.12	1.37	0.06	2.05	0.16	2.19	0.14	1.62	0.08	2.03	0.13	2.18	0.12	1.69	0.08
	SVM	1.85	0.14	1.78	0.12	1.56	0.11	1.16	0.08	1.80	0.12	1.66	0.12	1.25	0.09	1.77	0.12	1.62	0.10	1.23	0.08
3	OLS	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40	9.13	0.40
	AIC B	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40
	BIC B	9.07	0.40	9.07	0.40	9.07	0.40	9.08	0.40	9.08	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40
	AIC SB	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40
	BIC SB	9.07	0.40	9.07	0.40	9.07	0.40	9.08	0.40	9.08	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40
	AIC F	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.09	0.40	9.10	0.40	9.10	0.40	9.10	0.40
	BIC F	9.07	0.40	9.07	0.40	9.07	0.40	9.08	0.40	9.08	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40
	AIC SF	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.10	0.40	9.09	0.40	9.10	0.40	9.10	0.40	9.10	0.40
	BIC SF	9.07	0.40	9.07	0.40	9.07	0.40	9.08	0.40	9.08	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40	9.07	0.40
	Ridge	10.21	0.50	10.30	0.51	10.91	0.59	12.92	0.62	10.29	0.55	10.82	0.59	12.62	0.56	10.30	0.55	10.78	0.59	12.62	0.66
	Lasso	9.50	0.45	9.44	0.45	9.45	0.47	9.47	0.44	9.46	0.47	9.46	0.45	9.50	0.44	9.46	0.47	9.44	0.47	9.45	0.44
	E-net	9.48	0.44	9.43	0.44	9.44	0.46	9.46	0.42	9.45	0.45	9.45	0.45	9.49	0.43	9.44	0.46	9.43	0.46	9.45	0.43
	SCAD	9.08	0.40	9.08	0.40	9.08	0.40	9.08	0.40	9.08	0.39	9.08	0.39	9.08	0.40	9.08	0.40	9.08	0.40	9.08	0.40
	MCP	9.08	0.41	9.08	0.40	9.08	0.40	9.08	0.41	9.08	0.39	9.08	0.39	9.08	0.40	9.08	0.40	9.08	0.40	9.09	0.40
	XGBoost	11.01	0.60	10.98	0.53	10.89	0.54	11.16	0.78	11.00	0.54	10.97	0.53	11.16	0.77	10.96	0.59	10.98	0.57	11.08	0.83
	RF	18.32	1.33	18.35	1.12	17.22	0.99	12.38	0.58	18.25	1.43	19.68	1.31	14.60	0.69	18.40	1.40	19.47	1.18	15.10	0.68
	SVM	16.69	1.28	15.84	0.99	13.96	0.93	10.46	0.77	16.21	1.12	14.96	1.09	11.24	0.77	16.03	1.20	14.40	0.91	11.09	0.67
6	OLS	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59	36.50	1.59
	AIC B	36.41	1.60	36.41	1.61	36.41	1.60	36.39	1.61	36.41	1.60	36.42	1.59	36.39	1.62	36.39	1.59	36.41	1.61	36.40	1.60
	BIC B	36.28	1.60	36.26	1.62	36.28	1.59	36.26	1.59	36.30	1.58	36.29	1.58	36.29	1.61	36.29	1.60	36.28	1.60	36.28	1.61
	AIC SB	36.41	1.60	36.41	1.61	36.41	1.60	36.39	1.61	36.41	1.60	36.42	1.59	36.39	1.62	36.39	1.59	36.41	1.61	36.40	1.60
	BIC SB	36.28	1.60	36.26	1.62	36.28	1.59	36.26	1.59	36.30	1.58	36.29	1.58	36.29	1.61	36.29	1.60	36.28	1.60	36.28	1.61
	AIC F	36.41	1.60	36.41	1.61	36.40	1.60	36.39	1.61	36.41	1.60	36.41	1.59	36.37	1.60	36.40	1.59	36.40	1.61	36.39	1.61
	BIC F	36.28	1.60	36.26	1.62	36.28	1.59	36.26	1.59	36.30	1.58	36.28	1.59	36.28	1.62	36.29	1.60	36.28	1.60	36.28	1.61
	AIC SF	36.41	1.60	36.41	1.61	36.40	1.60	36.39	1.61	36.41	1.60	36.41	1.59	36.37	1.60	36.40	1.59	36.40	1.61	36.39	1.61
	BIC SF	36.28	1.60	36.26	1.62	36.28	1.59	36.26	1.59	36.30	1.58	36.28	1.59	36.28	1.62	36.29	1.60	36.28	1.60	36.28	1.61
	Ridge	40.85	2.02	41.21	2.02	43.66	2.34	51.70	2.49	41.15	2.18	43.27	2.36	50.50	2.23	41.19	2.21	43.13	2.37	50.49	2.66
	Lasso	37.99	1.80	37.77	1.81	37.82	1.88	37.89	1.77	37.85	1.90	37.85	1.78	37.99	1.76	37.83	1.86	37.76	1.88	37.81	1.75
	E-net	37.93	1.76	37.73	1.75	37.76	1.82	37.82	1.68	37.78	1.78	37.81	1.78	37.95	1.72	37.77	1.83	37.73	1.85	37.78	1.74
	SCAD	36.31	1.60	36.31	1.61	36.32	1.58	36.32	1.60	36.30	1.58	36.34	1.58	36.34	1.61	36.32	1.59	36.33	1.59	36.33	1.59
	MCP	36.33	1.62	36.31	1.61	36.32	1.59	36.30	1.63	36.32	1.57	36.32	1.57	36.33	1.61	36.33	1.60	36.33	1.60	36.34	1.60
	XGBoost	44.06	2.40	43.89	2.11	43.61	2.18	44.52	2.85	43.94	2.08	43.87	2.18	44.68	3.13	43.86	2.42	43.88	2.17	44.06	2.77
	RF	73.27	5.35	73.37	4.50	68.90	3.97	49.51	2.33	73.02	5.72	78.72	5.21	98.40	2.75	73.60	5.60	77.88	4.72	60.38	2.71
	SVM	66.76	5.12	63.36	3.98	55.88	3.75	41.80	2.98	64.85	4.47	59.86	4.36	44.94	3.12	63.99	4.57	57.67	3.69	44.39	2.74

Table 6: Mean and standard deviation of the β-sensitivity when $n = 1000$ and $p = 10$. See Figure **??** for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	OLS	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC B	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC B	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC SB	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC SB	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC F	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC F	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC SF	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC SF	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	Ridge	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	Lasso	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0.998	0.02
	E-net	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0.998	0.02
	SCAD	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	MCP	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
3	OLS	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC B	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC B	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC SB	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC SB	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC F	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC F	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC SF	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC SF	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	Ridge	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	Lasso	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0.998	0.02
	E-net	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0.998	0.02
	SCAD	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	MCP	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
6	OLS	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC B	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC B	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC SB	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC SB	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC F	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC F	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	AIC SF	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	BIC SF	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	Ridge	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	Lasso	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0.998	0.02
	E-net	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0.998	0.02
	SCAD	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00
	MCP	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1.000	0.00

Table 7: Mean and standard deviation of the β-specificity when $n = 1000$ and $p = 10$. See Figure **??** for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	OLS	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	AIC B	0.8317	0.1526	0.8333	0.1607	0.8267	0.1640	0.8367	0.1589	0.8450	0.1503	0.8100	0.1675	0.8050	0.1742	0.8217	0.1594	0.8067	0.1736	0.8300	0.1641
	BIC B	0.9917	0.0365	0.9900	0.0398	0.9917	0.0365	0.9900	0.0398	0.9883	0.0427	0.9900	0.0571	0.9850	0.0585	0.9883	0.0489	0.9933	0.0328	0.9900	0.0398
	AIC SB	0.8317	0.1526	0.8333	0.1607	0.8267	0.1640	0.8367	0.1589	0.8450	0.1503	0.8100	0.1675	0.8050	0.1742	0.8217	0.1594	0.8067	0.1736	0.8283	0.1632
	BIC SB	0.9917	0.0365	0.9900	0.0398	0.9917	0.0365	0.9900	0.0398	0.9883	0.0427	0.9900	0.0571	0.9850	0.0585	0.9883	0.0489	0.9933	0.0328	0.9900	0.0398
	AIC F	0.8317	0.1526	0.8483	0.1443	0.8367	0.1606	0.8500	0.1544	0.8450	0.1503	0.8233	0.1514	0.8683	0.1387	0.8267	0.1570	0.8167	0.1683	0.8633	0.1369
	BIC F	0.9917	0.0365	0.9900	0.0398	0.9917	0.0365	0.9900	0.0398	0.9883	0.0427	0.9950	0.0286	0.9883	0.0489	0.9883	0.0489	0.9933	0.0328	0.9900	0.0398
	AIC SF	0.8317	0.1526	0.8483	0.1443	0.8367	0.1606	0.8500	0.1544	0.8450	0.1503	0.8233	0.1514	0.8683	0.1387	0.8267	0.1570	0.8167	0.1683	0.8633	0.1369
	BIC SF	0.9917	0.0365	0.9900	0.0398	0.9917	0.0365	0.9900	0.0398	0.9883	0.0427	0.9950	0.0286	0.9883	0.0489	0.9883	0.0489	0.9933	0.0328	0.9900	0.0398
	Ridge	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Lasso	0.9983	0.0167	0.9850	0.0535	0.9683	0.0775	0.9600	0.0754	0.9833	0.0691	0.9683	0.0738	0.8650	0.1492	0.9800	0.0639	0.9383	0.1102	0.8317	0.1562
	E-net	0.9883	0.0427	0.9783	0.0611	0.9467	0.1002	0.9117	0.1329	0.9750	0.0763	0.9333	0.1086	0.8167	0.1580	0.9733	0.0739	0.8917	0.1306	0.7450	0.1469
	SCAD	0.9000	0.2145	0.9017	0.2146	0.9133	0.2228	0.9167	0.1975	0.8850	0.2256	0.8617	0.2763	0.8583	0.2556	0.8850	0.2492	0.8817	0.2510	0.9217	0.2002
	MCP	0.8983	0.2144	0.9283	0.1869	0.9100	0.2228	0.9133	0.2085	0.8900	0.2238	0.9250	0.2137	0.8817	0.2069	0.9100	0.2124	0.8900	0.2453	0.9200	0.2017
3	OLS	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	AIC B	0.8317	0.1526	0.8033	0.1698	0.8167	0.1683	0.8467	0.1583	0.8467	0.1473	0.8100	0.1675	0.8200	0.1934	0.8383	0.1666	0.8183	0.1726	0.8300	0.1624
	BIC B	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9900	0.0571	0.9850	0.0631	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	AIC SB	0.8317	0.1526	0.8033	0.1698	0.8167	0.1683	0.8467	0.1583	0.8467	0.1473	0.8100	0.1675	0.8183	0.1926	0.8383	0.1666	0.8183	0.1726	0.8300	0.1624
	BIC SB	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9900	0.0571	0.9850	0.0631	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	AIC F	0.8317	0.1526	0.8033	0.1698	0.8033	0.1658	0.8533	0.1503	0.8500	0.1431	0.8233	0.1514	0.8717	0.1399	0.8417	0.1665	0.8400	0.1640	0.8550	0.1529
	BIC F	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9950	0.0286	0.9917	0.0435	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	AIC SF	0.8317	0.1526	0.8033	0.1698	0.8333	0.1658	0.8533	0.1503	0.8500	0.1431	0.8233	0.1514	0.8717	0.1399	0.8417	0.1665	0.8400	0.1640	0.8550	0.1529
	BIC SF	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9950	0.0286	0.9917	0.0435	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	Ridge	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Lasso	0.9983	0.0167	0.9850	0.0535	0.9733	0.0811	0.9467	0.1056	0.9917	0.0365	0.9683	0.0738	0.8717	0.1457	0.9800	0.0682	0.9467	0.0914	0.8300	0.1381
	E-net	0.9883	0.0427	0.9717	0.0672	0.9467	0.1082	0.9017	0.1403	0.9867	0.0612	0.9333	0.1086	0.8150	0.1605	0.9683	0.0877	0.9017	0.1256	0.7400	0.1523
	SCAD	0.9000	0.2145	0.8617	0.2783	0.8917	0.2409	0.9083	0.2043	0.9133	0.1931	0.8617	0.2763	0.8883	0.2159	0.8850	0.2749	0.8900	0.2498	0.9033	0.2250
	MCP	0.8983	0.2144	0.9083	0.2176	0.9133	0.2017	0.9317	0.1626	0.9350	0.1533	0.9250	0.2137	0.9050	0.2096	0.9000	0.2404	0.9200	0.2044	0.9133	0.1946
6	OLS	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	AIC B	0.8317	0.1526	0.8033	0.1698	0.8167	0.1683	0.8467	0.1583	0.8467	0.1473	0.8100	0.1675	0.8200	0.1934	0.8383	0.1666	0.8183	0.1726	0.8300	0.1624
	BIC B	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9900	0.0571	0.9850	0.0631	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	AIC SB	0.8317	0.1526	0.8033	0.1698	0.8167	0.1683	0.8467	0.1583	0.8467	0.1473	0.8100	0.1675	0.8183	0.1926	0.8383	0.1666	0.8183	0.1726	0.8300	0.1624
	BIC SB	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9900	0.0571	0.9850	0.0631	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	AIC F	0.8317	0.1526	0.8033	0.1698	0.8033	0.1658	0.8533	0.1503	0.8500	0.1431	0.8233	0.1514	0.8717	0.1399	0.8417	0.1665	0.8400	0.1640	0.8550	0.1529
	BIC F	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9950	0.0286	0.9917	0.0435	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	AIC SF	0.8317	0.1526	0.8033	0.1698	0.8333	0.1658	0.8533	0.1503	0.8500	0.1431	0.8233	0.1514	0.8717	0.1399	0.8417	0.1665	0.8400	0.1640	0.8550	0.1529
	BIC SF	0.9917	0.0365	0.9900	0.0463	0.9883	0.0489	0.9867	0.0454	0.9900	0.0398	0.9950	0.0286	0.9917	0.0435	0.9883	0.0489	0.9917	0.0365	0.9917	0.0365
	Ridge	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Lasso	0.9983	0.0167	0.9850	0.0535	0.9733	0.0811	0.9467	0.1056	0.9917	0.0365	0.9683	0.0738	0.8717	0.1457	0.9800	0.0682	0.9467	0.0914	0.8300	0.1381
	E-net	0.9883	0.0427	0.9717	0.0672	0.9467	0.1082	0.9017	0.1403	0.9867	0.0612	0.9333	0.1086	0.8150	0.1605	0.9683	0.0877	0.9017	0.1256	0.7400	0.1523
	SCAD	0.9000	0.2145	0.8617	0.2783	0.8917	0.2409	0.9083	0.2043	0.9133	0.1931	0.8617	0.2763	0.8883	0.2159	0.8850	0.2749	0.8900	0.2498	0.9033	0.2250
	MCP	0.8983	0.2144	0.9083	0.2176	0.9133	0.2017	0.9317	0.1626	0.9350	0.1533	0.9250	0.2137	0.9050	0.2096	0.9000	0.2404	0.9200	0.2044	0.9133	0.1946

Table 8: Mean and standard deviation of the mean squared error on training data when $n = 50$ and $p = 2000$. See Figure ?? for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	Lasso	2.01	0.84	1.75	1.10	1.81	1.04	1.60	0.48	2.45	1.62	4.61	2.38	2.06	0.74	1.86	1.15	3.32	1.75	1.84	0.47
	E-net	2.26	1.34	2.15	1.42	2.19	1.31	1.54	0.45	2.93	1.83	4.71	2.25	2.16	0.74	2.51	1.81	3.64	1.83	1.87	0.47
	SCAD	0.77	0.34	0.70	0.33	0.84	0.28	1.30	0.43	0.74	0.33	1.11	1.11	1.40	0.56	0.72	0.32	0.91	0.56	1.45	0.47
	MCP	0.87	0.33	0.89	0.34	0.96	0.36	1.26	0.42	0.88	0.27	1.59	1.31	1.53	0.48	0.90	0.30	1.10	0.63	1.51	0.46
	XGBoost	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	RF	2.14	0.40	1.91	0.38	1.31	0.23	0.45	0.10	1.94	0.43	1.48	0.33	0.61	0.14	1.87	0.41	1.26	0.25	0.51	0.12
	SVM	3.75	3.59	2.65	2.83	1.23	1.44	1.01	0.61	4.13	3.75	3.74	3.26	0.87	1.38	2.69	3.15	0.68	1.02	0.20	0.11
3	Lasso	18.11	7.55	18.29	12.59	14.71	9.87	13.78	4.35	21.47	14.38	37.19	21.15	19.11	6.00	22.42	16.17	26.68	17.69	17.06	3.80
	E-net	20.35	12.10	19.87	12.92	15.86	10.46	13.23	4.64	25.37	17.20	40.30	21.89	19.82	6.51	26.27	18.03	32.27	18.12	17.30	4.35
	SCAD	6.94	3.08	6.95	3.59	7.56	2.44	12.66	3.68	6.59	2.96	9.43	8.15	12.56	4.93	7.22	3.21	7.45	4.30	13.44	3.94
	MCP	7.84	2.93	7.93	2.75	8.43	2.25	11.63	4.00	7.70	2.57	15.04	11.94	13.80	4.18	8.44	3.69	9.52	6.32	14.18	3.40
	XGBoost	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	RF	19.31	3.60	16.96	3.35	11.94	2.35	4.09	0.85	17.37	3.90	13.25	2.84	5.55	1.23	17.03	3.06	11.41	2.25	4.79	0.91
	SVM	34.19	30.66	22.00	24.86	11.83	15.63	8.12	4.04	36.17	32.02	28.73	29.54	9.09	13.91	20.86	25.79	6.75	9.91	1.69	0.66
6	Lasso	72.43	30.22	73.15	50.35	58.84	39.48	55.14	17.39	90.05	67.36	148.78	84.60	76.43	24.00	89.69	64.66	106.72	70.77	68.24	15.20
	E-net	81.40	48.39	79.47	51.68	63.43	41.85	52.93	18.57	102.83	64.68	161.21	87.58	79.27	26.05	105.07	72.12	129.10	72.49	69.20	17.40
	SCAD	27.77	12.34	27.82	14.35	30.26	9.76	50.65	14.71	28.49	11.19	37.71	32.58	50.22	19.72	28.88	12.84	29.80	17.22	53.75	15.75
	MCP	31.37	11.71	31.72	10.98	33.72	9.00	46.51	16.00	32.96	15.41	60.14	47.76	55.20	16.70	33.78	14.76	38.07	25.28	56.70	13.61
	XGBoost	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	RF	77.31	14.37	67.78	13.35	47.85	9.60	16.38	3.44	68.81	15.73	53.11	11.43	22.23	4.96	68.02	12.31	45.69	8.92	19.15	3.69
	SVM	159.93	133.99	81.76	94.03	49.00	59.08	33.34	19.99	142.86	132.04	112.23	116.37	49.15	73.90	88.08	105.63	28.52	47.94	6.74	2.65

Table 9: Mean and standard deviation of the mean squared error on training data when $n = 50$ and $p = 2000$. See Figure ?? for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	Lasso	3.29	1.57	3.30	2.17	3.45	1.61	2.45	0.71	4.07	2.96	5.87	2.27	2.55	0.70	3.47	1.74	5.09	2.24	2.34	0.75
	E-net	4.01	2.40	4.04	2.55	4.11	1.86	2.52	0.74	4.95	3.45	6.22	2.20	2.72	0.81	4.39	2.35	5.56	2.34	2.45	0.76
	SCAD	1.31	0.28	1.36	0.31	1.28	0.34	2.00	0.72	1.33	0.38	2.36	1.93	1.97	0.42	1.36	0.31	1.68	1.15	2.07	0.59
	MCP	1.33	0.30	1.36	0.33	1.43	0.91	1.92	0.74	1.32	0.35	2.60	2.07	1.94	0.39	1.39	0.37	2.02	1.76	2.13	0.62
	XGBoost	13.34	4.78	12.06	3.79	9.56	2.38	3.56	0.95	12.43	3.65	9.34	2.48	4.06	1.35	11.74	2.83	8.94	2.80	3.42	0.84
	RF	15.01	3.80	12.93	3.20	9.47	1.82	3.16	0.77	13.32	3.61	9.73	2.12	4.25	1.42	12.51	2.81	9.31	2.44	3.41	0.93
	SVM	18.18	4.06	15.75	3.21	11.37	2.32	4.12	1.71	17.60	3.63	15.29	2.61	12.26	2.68	16.97	3.33	13.97	2.73	7.66	1.95
3	Lasso	29.59	14.14	32.13	18.61	29.27	13.88	21.48	7.12	36.38	27.60	50.59	20.76	23.12	6.66	37.10	23.41	41.95	17.77	20.23	5.78
	E-net	36.11	21.63	37.22	20.50	33.26	15.88	22.04	7.40	44.04	33.05	55.61	20.21	24.65	7.24	44.32	25.21	47.57	17.17	21.58	6.23
	SCAD	11.81	2.53	12.32	3.90	11.26	2.58	20.18	8.37	12.23	3.32	20.29	17.06	17.45	3.60	12.99	7.60	15.06	12.71	18.63	5.29
	MCP	11.99	2.71	12.38	3.71	11.33	2.54	19.00	7.35	12.07	3.38	24.41	19.61	17.54	3.77	13.03	7.45	15.20	11.86	19.35	5.96
	XGBoost	117.20	39.53	109.72	31.28	81.61	22.71	31.49	8.64	110.46	31.26	84.20	22.66	35.56	10.33	104.68	28.43	75.11	18.49	30.93	9.38
	RF	135.18	34.03	116.37	27.61	82.55	19.63	27.53	7.30	119.28	31.01	87.64	19.98	37.79	12.42	112.50	27.47	79.84	19.08	29.90	9.42
	SVM	163.73	36.63	145.13	30.24	101.39	22.45	35.85	14.32	157.70	33.62	137.34	23.79	109.13	24.94	149.19	30.47	125.37	26.07	66.95	15.98
6	Lasso	118.35	56.56	128.51	74.46	117.09	55.54	85.93	28.47	149.85	112.89	202.37	83.04	92.50	26.63	148.40	93.64	167.81	71.09	80.90	23.12
	E-net	144.43	86.54	148.87	82.01	133.02	63.52	88.15	29.61	178.74	126.56	222.45	80.86	98.59	28.97	177.28	100.85	190.29	68.67	86.33	24.92
	SCAD	47.25	10.14	49.26	15.59	45.05	10.33	80.73	33.47	48.21	13.95	81.17	68.26	69.82	14.39	51.96	30.40	60.25	50.84	74.52	21.16
	MCP	47.96	10.83	49.51	14.84	45.32	10.14	75.98	29.39	49.06	17.54	97.64	78.45	70.14	15.10	52.12	29.79	60.81	47.45	77.41	23.86
	XGBoost	462.42	153.08	438.75	132.30	329.77	90.89	125.16	35.21	446.56	135.46	344.83	94.89	143.48	43.70	420.52	121.39	301.37	73.14	124.61	35.86
	RF	540.97	136.58	466.03	111.18	330.46	78.76	110.13	29.18	474.40	128.25	351.13	80.23	151.07	49.86	450.25	110.50	319.58	77.19	119.52	37.83
	SVM	655.71	146.59	581.13	117.87	406.59	89.66	143.00	57.08	629.70	132.87	549.67	94.82	437.80	101.11	598.30	121.48	502.02	104.82	267.80	63.93

Table 10: Mean and standard deviation of the β-sensitivity when $n = 50$ and $p = 2000$. See Figure **??** for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	Lasso	0.834	0.0755	0.834	0.1103	0.782	0.1140	0.548	0.1352	0.842	0.1401	0.574	0.2023	0.570	0.1374	0.830	0.1000	0.680	0.1504	0.578	0.0894
	E-net	0.828	0.0854	0.824	0.1182	0.760	0.1206	0.566	0.1174	0.834	0.1634	0.604	0.2089	0.674	0.1088	0.806	0.1118	0.668	0.1399	0.618	0.0903
	SCAD	0.900	0.1005	0.896	0.1004	0.832	0.0737	0.500	0.1570	0.910	0.1000	0.766	0.1730	0.416	0.0545	0.862	0.0930	0.788	0.1200	0.432	0.0973
	MCP	0.864	0.0938	0.852	0.0882	0.804	0.0803	0.504	0.1669	0.886	0.0995	0.712	0.1871	0.406	0.0343	0.824	0.0653	0.748	0.1521	0.414	0.0711
3	Lasso	0.834	0.0755	0.812	0.1018	0.794	0.1188	0.542	0.1372	0.836	0.1375	0.626	0.2196	0.564	0.1404	0.806	0.1347	0.706	0.1406	0.606	0.0874
	E-net	0.828	0.0854	0.804	0.1024	0.792	0.1169	0.568	0.1325	0.832	0.1576	0.634	0.2203	0.670	0.1115	0.794	0.1462	0.680	0.1303	0.644	0.0925
	SCAD	0.900	0.1005	0.880	0.0985	0.828	0.0697	0.450	0.1460	0.900	0.1005	0.782	0.1533	0.414	0.0513	0.860	0.1005	0.814	0.1110	0.412	0.0556
	MCP	0.864	0.0938	0.846	0.0846	0.812	0.0477	0.464	0.1501	0.872	0.0965	0.704	0.1831	0.408	0.0394	0.836	0.0959	0.786	0.1146	0.400	0.0402
6	Lasso	0.834	0.0755	0.812	0.1018	0.794	0.1188	0.542	0.1372	0.838	0.1469	0.626	0.2196	0.564	0.1404	0.806	0.1347	0.706	0.1406	0.606	0.0874
	E-net	0.828	0.0854	0.804	0.1024	0.792	0.1169	0.568	0.1325	0.840	0.1449	0.634	0.2203	0.670	0.1115	0.794	0.1462	0.680	0.1303	0.644	0.0925
	SCAD	0.900	0.1005	0.880	0.0985	0.828	0.0697	0.450	0.1460	0.902	0.1005	0.782	0.1533	0.414	0.0513	0.860	0.1005	0.814	0.1110	0.412	0.0556
	MCP	0.864	0.0938	0.846	0.0846	0.812	0.0477	0.464	0.1501	0.868	0.0952	0.704	0.1831	0.408	0.0394	0.836	0.0959	0.786	0.1146	0.400	0.0402

Table 11: Mean and standard deviation of the β-specificity when $n = 50$ and $p = 2000$. See Figure **??** for the corresponding plot.

	Type	Independent		Symmetric						Autoregressive						Blockwise					
	Corr.	0		0.2		0.5		0.9		0.2		0.5		0.9		0.2		0.5		0.9	
σ	Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	Lasso	0.9966	0.0031	0.9950	0.0035	0.9947	0.0032	0.9959	0.0023	0.9966	0.0027	0.9978	0.0030	0.9986	0.0031	0.9952	0.0034	0.9968	0.0039	0.9986	0.0013
	E-net	0.9955	0.0034	0.9945	0.0034	0.9943	0.0034	0.9929	0.0026	0.9960	0.0029	0.9973	0.0033	0.9984	0.0024	0.9952	0.0034	0.9965	0.0042	0.9969	0.0015
	SCAD	0.9966	0.0034	0.9959	0.0035	0.9981	0.0023	0.9994	0.0013	0.9962	0.0035	0.9952	0.0039	0.9975	0.0035	0.9961	0.0035	0.9967	0.0033	0.9986	0.0020
	MCP	0.9991	0.0013	0.9991	0.0012	0.9996	0.0006	0.9997	0.0004	0.9990	0.0010	0.9990	0.0012	0.9994	0.0011	0.9992	0.0013	0.9990	0.0013	0.9994	0.0012
3	Lasso	0.9966	0.0031	0.9955	0.0031	0.9942	0.0036	0.9959	0.0023	0.9963	0.0029	0.9972	0.0033	0.9990	0.0018	0.9960	0.0030	0.9955	0.0051	0.9988	0.0012
	E-net	0.9955	0.0034	0.9946	0.0032	0.9932	0.0037	0.9928	0.0027	0.9956	0.0027	0.9967	0.0039	0.9985	0.0022	0.9952	0.0033	0.9960	0.0052	0.9970	0.0017
	SCAD	0.9966	0.0034	0.9965	0.0032	0.9979	0.0024	0.9991	0.0015	0.9963	0.0037	0.9956	0.0035	0.9976	0.0033	0.9967	0.0034	0.9964	0.0033	0.9986	0.0021
	MCP	0.9991	0.0013	0.9991	0.0012	0.9997	0.0006	0.9997	0.0005	0.9990	0.0012	0.9990	0.0013	0.9994	0.0010	0.9991	0.0012	0.9993	0.0010	0.9995	0.0009
6	Lasso	0.9966	0.0031	0.9955	0.0031	0.9942	0.0036	0.9959	0.0023	0.9965	0.0029	0.9972	0.0033	0.9990	0.0018	0.9960	0.0030	0.9955	0.0051	0.9988	0.0012
	E-net	0.9955	0.0034	0.9946	0.0032	0.9932	0.0037	0.9928	0.0027	0.9957	0.0030	0.9967	0.0039	0.9985	0.0022	0.9952	0.0033	0.9960	0.0052	0.9970	0.0017
	SCAD	0.9966	0.0034	0.9965	0.0032	0.9979	0.0024	0.9991	0.0015	0.9970	0.0032	0.9956	0.0035	0.9976	0.0033	0.9967	0.0034	0.9964	0.0033	0.9986	0.0021
	MCP	0.9991	0.0013	0.9991	0.0012	0.9997	0.0006	0.9997	0.0005	0.9991	0.0012	0.9990	0.0013	0.9994	0.0010	0.9991	0.0012	0.9993	0.0010	0.9995	0.0009