A Comparative Study of Penalized Regression and Machine Learning Algorithms in High Dimensional Scenarios

Gabriel Ackall[†] and Connor Shrader[‡]

Project advisor: Seongtae Kim[§]

Abstract. With the prevalence of big data in recent years, the importance of modeling high dimensional data and selecting important features has increased greatly. High dimensional data is common in many fields such as genome decoding, rare disease identification, 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, elastic net, smoothly clipped absolute deviation, and minimax concave penalty compared to traditional machine learning models such as random forest, XGBoost, and support vector machines. We compare these models using factorial design methods for Monte Carlo simulations in 540 environments, with factors being the response variable, 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. We evaluate the models using the training and test mean squared error, variable selection accuracy, β-sensitivity, and β-specificity. We found that the performance of penalized regression models is comparable with traditional machine learning algorithms in most high-dimensional situations. 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. Our study shows that penalized regression techniques should be included in predictive modelers' toolbox.

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

[†]Georgia Institute of Technology, Civil Engineering, Atlanta, GA (gackall@gatech.edu).

[‡]University of Central Florida, Mathematics, Orlando, FL (connorshrader@knights.ucf.edu).

[§]North Carolina A&T State University, Mathematics and Statistics, Greensboro, NC

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 [30]. This is due in part to tree based models such as XGBoost winning numerous machine learning competitions [30]. 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. Statistical methods lesser used by the machine learning community are sometimes able to perform equivalently or even better than more popular methods [24, 28]. However, there is a distinct lack of academia focusing on comparing these statistical, variable selection techniques with the increasingly popular machine learning techniques on a broad variety of data sets. This paper serves to help bridge that gap and promote the use of variable selection techniques within the greater machine learning community.

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; instead of finding generalized patterns for a population of data, they memorize specific occurrences in the training data [22, 16]. 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 high-dimensional data, including the large p, small n problem. *Variable selection* techniques overcome this issue by using only a subset of the available predictors. There are many ways to implement variable selection in models. We studied penalized regression models, such as ridge regression [21], least absolute shrinkage and selection operation (lasso) [34], elastic-net (e-net) [46], smoothly clipped absolute deviation (SCAD) [14], and minimax concave penalty (MCP) [43]. Besides ridge regression, these models simultaneously select important predictors and fit a linear model. Models that perform variable selection are suitable for applications such as genomics, where there are hundreds or thousands of predictors; see, for example, [36, 25].

We also evaluated the performance of several machine learning models: random forests (RF) [5], gradient boosting in the form of XGBoost [7], and support vector machine (SVM) models [8]. These types of models do not assume a linear relationship between a response and its predictors. This allows the machine learning models to have better predictive performance on data sets where the relationship between the response and its predictors is non-linear; on the other hand, this also makes the machine learning models more susceptible to overfitting. In some applications, a combination of machine learning and variable selection is employed [38]. For example, machine learning models often measure feature importance, which can be used to determine the variables that are most important for making predictions. However, feature importance scores are relative to each other and variable selection is not intrinsic to these models.

To compare these different techniques, models were trained and evaluated using both Monte Carlo simulations and empirical genomic data. We are particularly interested in understanding in the predictive performance of these models, so we evaluated the models using the mean squared error (MSE) on both training and test data. For the linear models fitted on simulated data, we also measured the β-sensitivity and β-specificity metrics, which evaluate the ability for these models to identify important predictors [27].

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 4 explains our empirical data analysis and results. Section 5 is a discussion of our results and Section 6 is the conclusion.

2. Methodology.

2.1. Modeling Background.

Consider a random variable Y and p predictor variables $X_1, X_2, \ldots, X_p$. This study focuses on *regression modeling*, where the response Y is a number on a continuous interval. We will assume that Y depends on some (or all) of the predictors; more specifically, we assume that

$$Y = f(X_1, X_2, \ldots, X_p) + \epsilon \tag{2.1}$$

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.

To compute $\hat{f}$, we use a *training set* of data. Suppose that we have a data set containing n observations of our p random variables. We will let x_{ij} denote the i-th observation of the j-th predictor variable, and we let y_i represent the i-th observation of the response.

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

$$f(X_1, X_2, \ldots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{2.2}$$

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

$$\text{RSS} = \sum_{i=1}^{n} \left[y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip}) \right]^2 \tag{2.3}$$

OLS is common because it is the best linear unbiased estimator; that is, OLS has a lower variance than any other linear unbiased estimator [19, 16]. 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.

Other types of linear models can overcome this large p small n problem by introducing a small amount of bias. In many cases, these models can perform *variable selection* by setting the coefficients of unimportant predictors to zero. There are several ways to implement variable selection into a linear model. *Filter methods* work by evaluating the ability for each individual predictors to predict the response; then, a model is fit using the predictors selected [32, 11]. *Wrapper methods* fit models using different subsets of predictors and choose the model that has the best performance [20, 27]. Finally, *embedded methods* perform variable selection during the model training process [20, 27]. This paper focuses on 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. 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 at the cost of introducing bias to the system. Most, but not all, of these methods can perform variable selection during the fitting process, making them a type of embedded method.

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

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \Big[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \Big]^2 + \sum_{j=1}^{p} P(\beta_j) \right\} \tag{2.4}$$

where the first summation is the usual residual sum of squares and $P(\beta)$ is a penalty function that is applied to each coefficient (not including the intercept β_0). This penalty usually depends on at least one tuning parameter (commonly denoted by λ) that controls how strong the penalty is. A suitable choice for the tuning parameter(s) will lead to a well-performing model.

Ridge regression is a penalized linear regression model that uses the penalty function $P(\beta) = \lambda\beta^2$, where $\lambda > 0$ is a tuning parameter [21]. Ridge regression benefits from having a closed-form solution that is easy to compute; it is also known for its ability to handle collinearity. However, unlike other models in this section, ridge regression is unable to perform variable selection.

The *least absolute shrinkage and selection operation* (lasso) is a shrinkage method with a very similar form to ridge regression [34, 22]. The penalty function for lasso is $P(\beta) = \lambda|\beta|$, where $\lambda > 0$. Like ridge regression, the lasso is a biased estimator. One significant advantage of lasso is that it can perform variable selection by setting coefficient estimates to zero.

Elastic-net (E-net) regression use the penalties of both ridge regression and the lasso [46]. Its penalty function is $P(\beta) = \lambda_1|\beta| + \lambda_2\beta^2$, where $\lambda_1, \lambda_2 > 0$ are separate tuning parameters. An equivalent way to express this penalty function is with $P(\beta) = \lambda((1-\alpha)\beta^2 + \alpha|\beta|)$, where

$\lambda > 0$ and $\alpha \in [0, 1]$ are tuning parameters. Note that if $\alpha = 0$, then the resulting model is just ridge regression, while using $\alpha = 1$ gives the lasso. The resulting model gains the advantages of both ridge regression and the lasso in that it can handle collinearity well while also performing variable selection.

The last two penalized linear models that we considered are *Smoothly-Clipped Absolute Deviation* (SCAD) and *Minimax Concave Penalty* (MCP) [14, 39, 43]. SCAD uses the penalty function

$$P(\beta) = \begin{cases} \lambda|\beta|, & |\beta| \leq \lambda \\ \frac{2a\lambda|\beta| - \beta^2 - \lambda^2}{(a-1)}, & \lambda \leq |\beta| < a\lambda \\ \frac{\lambda^2(a+1)}{2}, & a\lambda < |\beta| \end{cases} \tag{2.5}$$

while MCP uses

$$P(\beta) = \begin{cases} \lambda|\beta| - \frac{\beta^2}{2a}, & |\beta| \leq a\lambda \\ \frac{1}{2}a\lambda^2, & a\lambda < |\beta| \end{cases} \tag{2.6}$$

These methods use piecewise penalty functions that punish larger coefficients less severely. The resulting models are consequently the least biased methods among penalized regression models. Another feature of SCAD and MCP is their oracle-like properties [14, 43]. This means that as $n \to \infty$, SCAD and MCP will correctly identify exactly which predictors should have non-zero coefficients, and that their coefficient estimates will be normally distributed with the mean estimate being the true coefficient value [45].

2.3. 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, which will give the value that the decision tree predicts. 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; a small change in the training set can lead to a completely different decision tree [22].

A *random forest model* combines many independent decision trees to make one unified prediction [5]. Each tree is fitted independently using a random subset of observations chosen *with replacement* in a process called *bootstrapping* [13]. To make predictions using a random forest, the predictions for each individual decision tree are first computed. Then, the set of predictions are aggregated to give one final prediction. For regression, one suitable way to aggregate individual predictions is to use the mean.

Boosting is the technique of sequentially improving a weak learner until it becomes a strong learner [33]. Boosting is commonly used with decision trees. Unlike random forest

models, where each tree is independent of one another, the trees in a boosting model are fitted sequentially to correct the mistakes made by the previous tree. A *gradient boosting machine* (GBM) is a boosting technique that uses gradient descent to minimize error in a model and correct the shortcomings of previous iterations of the model [17].

The final non-linear model that we considered is the *support vector machine* (SVM) [8, 12]. Support vector machines find a hyperplane that closely fits the data. Data observations that are close to this hyperplane are called *support vectors*, and they have the strongest influence on the model. Unlike linear regression, support vector machines can address non-linear relationships between the response and its predictors.

2.4. Implementation. This section gives the specific details of how we fit each model for both the simulated data and the empirical data. Everything in our study was run on version 4.1.0 of `R` [31]. Table 1 summarizes the packages used for each model.

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

Library	Models used	Version
`stats` [31]	Ordinary least squares	4.1.0
`glmnet` [15]	Ridge, lasso, elastic-net	4.1-1
`ncvreg` [3]	SCAD and MCP	3.13.0
`xgboost` [7]	Gradient boosting	1.4.1.1
`ranger` [40]	Random forest (simulations)	0.12.1
`randomForest` [26]	Random forest (empirical data)	4.6-14
`e1071` [29]	Support vector machine	1.7-7

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

Ridge, lasso, and elastic-net models were fitted using `glmnet`. We used the `cv.glmnet` function, which uses cross validation grid search to optimize the penalty scalar λ. Cross validation helps generate a model that performs well on both training and testing data. We used the default value of 10 folds. For elastic-net regression, we found that the hyperparameter $\alpha = 0.8$ worked well in our simulation study and $\alpha = 0.5$ worked best in the empirical study. This means that the elastic-net model emphasizes the variable selection provided by lasso in the simulations. The remaining hyperparameters were given their default values.

We used the `cv.ncvreg` function from the `ncvreg` library for SCAD and MCP. 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 (the `ncvreg` documentation calls this parameter γ). All other arguments were given their default values.

For gradient boosting 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 a few 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 varied 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 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.

With random forests, we used `ranger` for the simulated data. For the empirical data, we instead used `randomForest` because `ranger` could not handle the large number of predictors, resulting in stack overflow errors. For both `ranger` and `randomForest`, we tuned the number of predictors considered per split in the learning process ($\lfloor\sqrt{p}\rfloor$, $\lfloor p/3\rfloor$, and $\lfloor p/2\rfloor$) and the number of trees (300, 400, 500 and 600). The minimum node size for each tree is 5. 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.

Some models used could only be used for certain values of n and p. This is because either the runtime becomes infeasible when n or p are large, or the model simply cannot be used when p is too large. Ordinary least squares was only used when $p \leq n$, since it cannot be used at all when $p > n$. 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 because support vector machine models could not handle such a high number of predictors in our empirical data.

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 additive linear and non-linear response functions, we obtain a more thorough understanding of how each model performs in different situations.

The additive linear response function assumes that

$$Y = 1 + 2X_1 - 2X_2 + 0.5X_5 + 3X_6 + \epsilon \tag{3.1}$$

where ϵ is an independent random error with mean 0 and constant variance. We refer to this linear response function as Model 1. Our additive non-linear response function uses

$$Y = 6 \times 1_{X_1>0} + X_2^2 + 0.5X_6 + 3X_7 + 2 \times 1_{X_8>0} \times 1_{X_9>0} + \epsilon \tag{3.2}$$

where $1_{X_i>0}$ is the index function given by

$$1_{X_i>0} = \begin{cases} 0, & X_i \leq 0 \\ 1, & X_i > 0 \end{cases} . \tag{3.3}$$

Note that the non-linear response still includes linear terms. We refer to this non-linear response function as Model 2.

For each simulation, we generated a random $n \times p$ matrix $\mathbf{X} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma})$, where $\mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma})$ is the multivariate normal distribution with p-dimensional mean vector $\mathbf{0}$ and $p \times p$ covariance matrix $\mathbf{\Sigma}$. The n-dimensional response vector $\mathbf{y}$ was then computed using one of the response functions described in Equations 3.1 and 3.2. Finally, the error term ϵ was generated from a normal distribution with mean 0 and variance σ^2, denoted as $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

We assumed that the variance for each predictor was 1, meaning that the covariance matrix $\mathbf{\Sigma}$ is actually a correlation matrix. For every $i \neq j$, the entry $\mathbf{\Sigma}_{ij} \in [0, 1]$ represents the correlation between predictors i and j. The diagonal entries are all equal to 1, indicating that each predictor has variance 1. Correlation between predictors can affect the ability for models to identify important predictors and make accurate predictions.

We considered the following correlation structures for our simulation study:

- *Independent* correlation, where $\mathbf{\Sigma}_{ij} = 0$ for all $i \neq j$;
- *Symmetric compound* correlation, where $\mathbf{\Sigma}_{ij} = \rho \in (0, 1)$ for all $i \neq j$;
- *Autoregressive correlation*, where $\mathbf{\Sigma}_{ij} = \rho^{|i-j|}$, where $\rho \in (0, 1)$; and
- *Blockwise correlation*, where $\mathbf{\Sigma}$ is block diagonal with each block having symmetric compound structure (also, each block uses the same value for ρ)

Our simulation study uses a *factorial design*, meaning that we ran simulations using every possible combination of different factors. The factors that we varied in our simulation study are

- The choice of response function (Model 1 or Model 2);
- 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 needed. 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.

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 using

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.4}$$

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 [27]. A coefficient estimate is a *true positive* (TP) 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* (TN) if the coefficient was correctly predicted to be zero when that predictor is not related to the response. A *false positive* (FP) happens when an important coefficient is incorrectly predicted to be non-zero. Finally, a coefficient estimate is a *false negative* (FN) 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.

The β-sensitivity of a model is given by

$$\beta\text{-sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.5}$$

while the β-specificity is given by

$$\beta\text{-specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.6}$$

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 of a model measures how well it can identify unimportant predictors (i.e. predictors that should be given a coefficient of zero).

3.3. Linear Simulation Results. Because we ran simulations using 540 different combinations of factors, we only show the results for $n = 50$ and $p = 2000$ in this report (representing the largest ratio between p and n). Results for other combinations of n and p are included in a supplementary document. We note that all results for ordinary least squares are shown in the supplementary document since we do not consider them when $p > n$.

Each plot measures the average value for one of the four metrics discussed above over 100 simulations. Each row of the plots represents a different value of σ, the standard deviation of

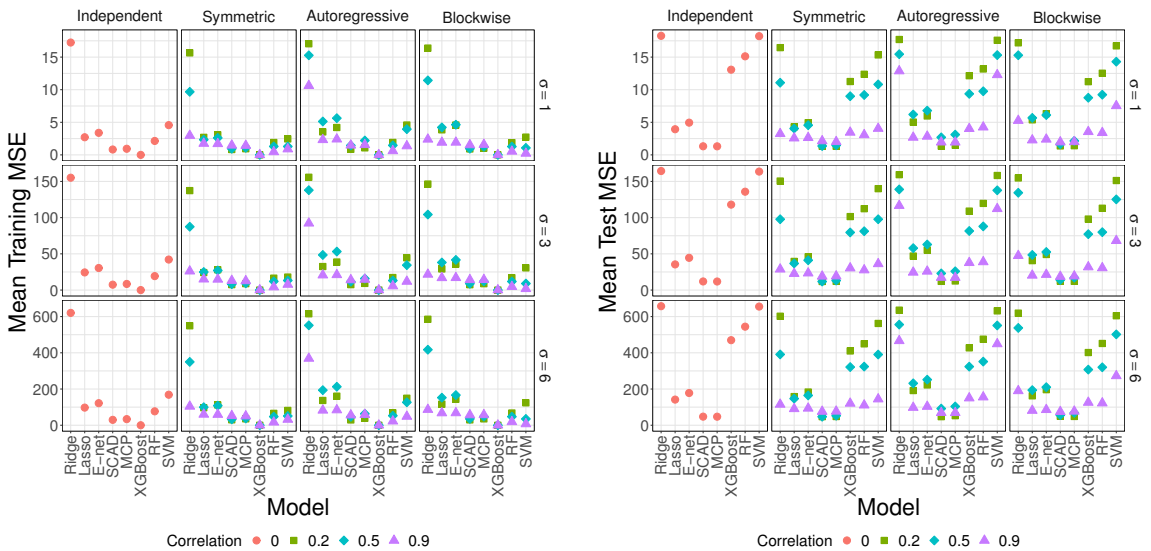
the random error. Each column represents a correlation structure. The different shapes and colors for each point represent the strength of the correlation between predictors.

We begin by presenting the results from our simulations for Model 1 (linear function), followed by the results from Model 2 (non-linear function).

Figure 1 shows the average MSE for the simulated models on both training data and test data. Figure 2 displays the β-sensitivity and β-specificity for the linear models that perform variable selection.

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. XGBoost 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.

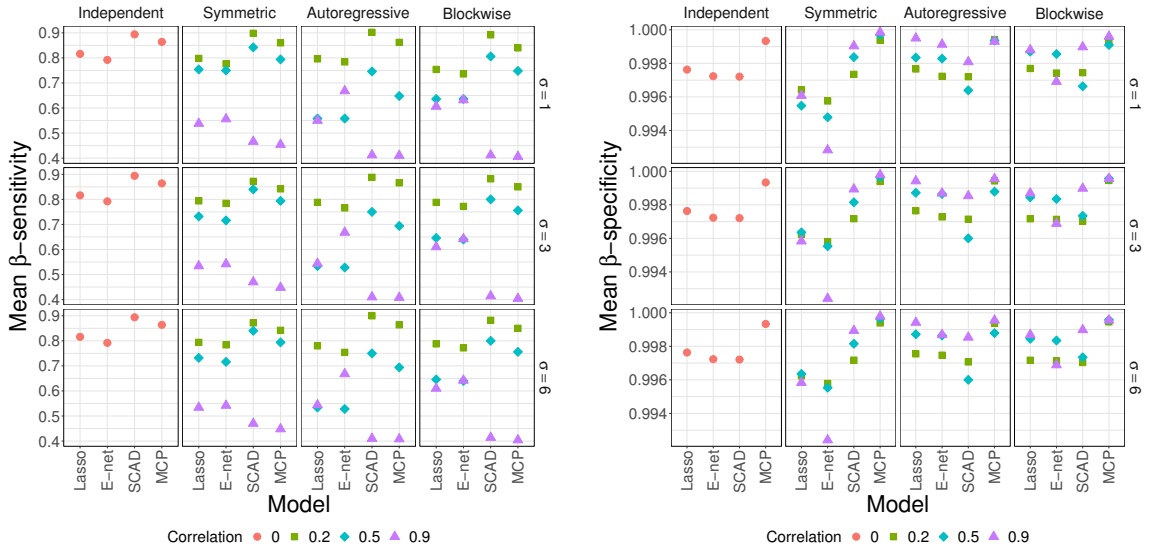
Looking at Figure 2a, we see that all of the models predict most of the non-zero coefficients when the correlation is low. When the correlation is high, all of the models struggle to identify the correct predictors. SCAD and MCP perform the best when the correlation is low but perform the worst when 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 autoregressive. The sensitivity of these models varies in importance depending on the situation where these models are utilized. In scenarios such as cancer detection where



(a) Training data.

(b) Test data.

Figure 1: Average mean squared error for linear simulations when $n = 50$ and $p = 2000$.



(a) β-sensitivity.

(b) β-specificity.

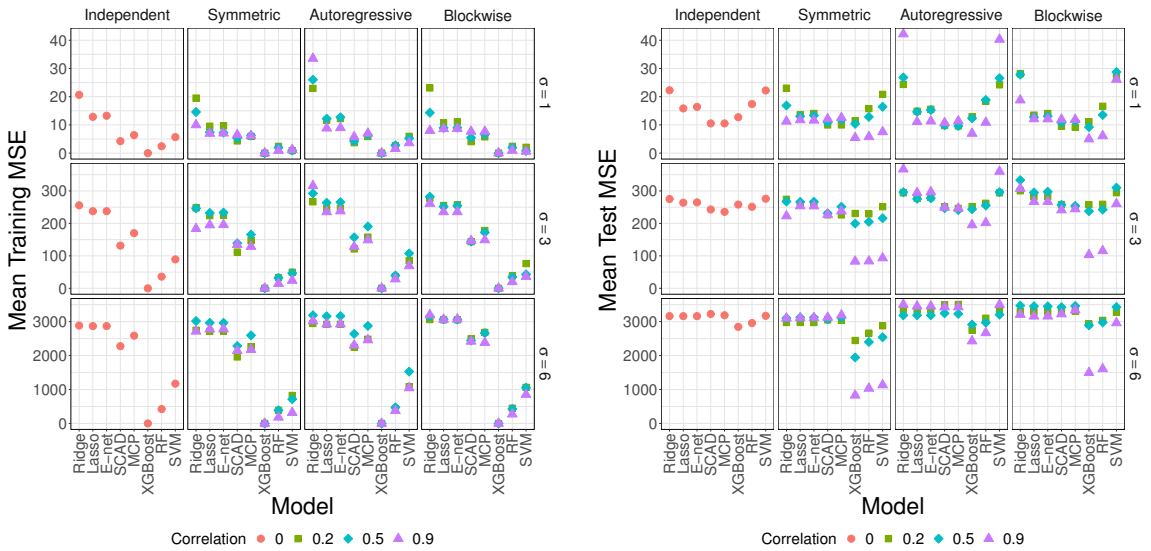
Figure 2: Average β-sensitivity and β-specificity for linear simulations when $n = 50$ and $p = 2000$.

the consequences of false negatives are large, a highly sensitive model is desired to reduce the amount of patients with cancer that are missed by an initial screening.

Now, consider the results for β-specificity from Figure 2 b. 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 correctly identify more coefficients as being zero as the correlation increases. Consider a similar scenario where models are used for diagnosing cancer patients. The specificity of a model proves much more useful in confirming results. Due to a high sensitivity model, patients can likely be falsely diagnosed positive, a highly specific model that then diagnoses this patient as positive can confirm the initial screening result. Thus, the specificity and sensitivity of a model are desired depending on the scenario they are applied to.

3.4. Non-linear Simulation Results. Now, we will highlight some results from the simulations for Model 2 given by Equation 3.2.

Figure 3 shows the average mean square errors on both the training data and test data. We see that the linear and non-linear models have similar test mean squared errors at medium levels of noise and correlation. Among the linear models, SCAD and MCP penalized regression techniques perform best, and at medium and low correlation levels perform better than non-linear models such as random forest and XGBoost. The non-linear models all have a noticeably lower test mean squared error when the correlation between predictors is high and perform



(a) Training data.

(b) Test data.

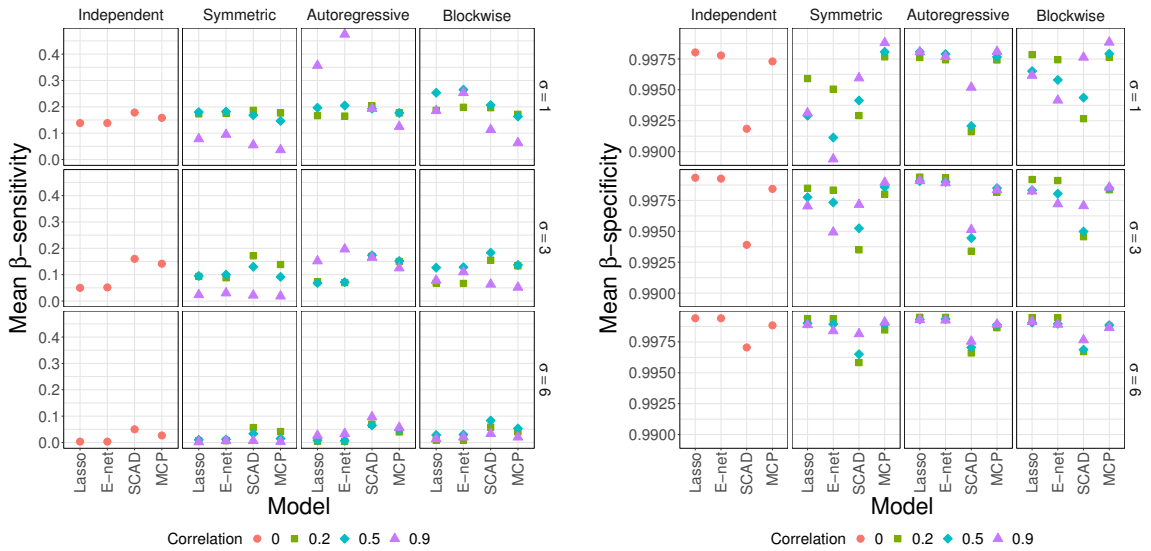
Figure 3: Average mean squared error for non-linear simulations when $n = 50$ and $p = 2000$.

better than any of the linear models at these high correlations. The non-linear models perform significantly better on training data than on test data. This is indicative of overfitting and on more complex data or variable scenarios this could decrease the accuracy of the non-linear models.

Figure 4 shows the results for the β-sensitivity and β-specificity for the non-linear simulations. 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. These results identify a key weakness of penalized regression methods, for more complex, non-linear simulations, the sensitivity of penalized regression models are very low, meaning they miss many significant predictors. On the other hand, the specificity of penalized regression models is very high, indicating that predictors that chosen by the models are very likely due to be significant. While it may be possible to increase the sensitivity of penalized regression models, this will likely come at the cost of specificity. Depending on the scenario, sensitivity may be more desired and thus this is a acceptable tradeoff.

4. Empirical Data Analysis.

4.1. 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 [42]. This data set contains the gene expression data of 17323 genes from



(a) β-sensitivity.

(b) β-specificity.

Figure 4: Average β-sensitivity and β-specificity for non-linear simulations when $n = 50$ and $p = 2000$.

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 [23, 1]. 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 [10]. Because the BRCA1 gene interacts with other genes, it is useful to find genes that interact with BRCA1 to test in further studies [10]. The variable selection capabilities of penalized regression techniques prove very useful in this application for narrowing large number of possible genes to more manageable numbers of genes more significant t o cancer development that can be studied in a laboratory and targeted to create cancer treatments. 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 used. This is because there are more predictors than samples which makes least squares linear regression impossible. Additionally, support vector machines struggle at such a high number of predictors and resulted in stack overflow e rrors w hich m ade fitting support vector machines on this data impossible. It is also important to note that we do not know whether the response variable is related linearly or non-linearly to the gene expression data. This is why it is important to analyze real, empirical data when comparing machine learning techniques since we cannot know the functional form of the data.

To evaluate the models, we used *nested cross validation*. We first s plit t he d ata i nto 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 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.

4.2. 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 5 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.

For lasso, elastic-net, MCP, and random forests, we also tracked the most important predictors selected by each model on each fold. For lasso, elastic-net, and MCP, we consider

Table 2: Train MSE, test MSE, and runtime metrics for models fit using the bcTCGA data set.

	Train MSE		**Test MSE**		
Model	**Mean**	**SD**	**Mean**	**SD**	**Mean Runtime** (s)
Ridge	0.1391	0.0266	0.2858	0.0610	29.29
Lasso	0.1842	0.0159	0.2304	0.0337	9.57
E-net	0.1799	0.0127	0.2281	0.0469	**9.41**
SCAD	0.1442	0.0333	0.2218	0.0303	17.28
MCP	0.1566	0.0129	**0.2202**	**0.0224**	15.15
GBM	**0.0002**	**0.0004**	0.2233	0.0507	538.12
RF	0.0378	0.0013	0.2653	0.0525	4906.59

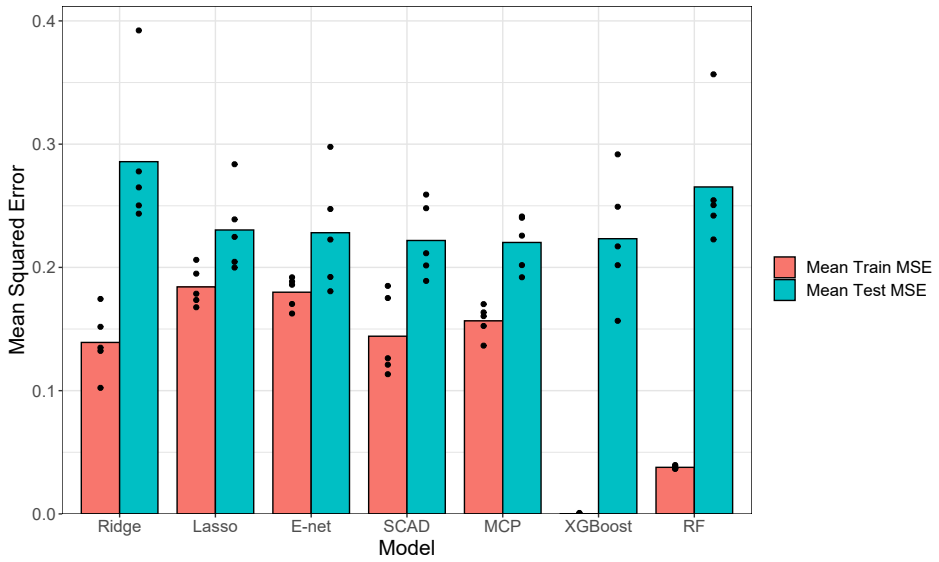


Figure 5: Mean squared error of the models fit on the bcTCGA data set. Each point represents the mean squared error for one fold, while the bars represent the average for the five folds.

a predictor to be important if it was assigned a non-zero coefficient. For random forests, each predictor is given an importance score by the algorithms we used; we considered the 50 predictors with the highest importance for each method. The important predictors that all four models had in common are listed below.

- Fold 1: DTL, NBR2, NPR2, VPS25
- Fold 2: DTL, KHDRBS1, NBR2, VPS25
- Fold 3: DTL, NBR2, VPS25
- Fold 4: NBR2
- Fold 5: NBR2, TIMELESS.

Tables 3 through 6 list the top five predictors chosen by each model on each cross-validation fold.

5. Discussion. We first describe our key findings from the simulation study. Although results were only shown for the case when $n = 50$ and $p = 2000$, some broad conclusions from other combinations of n and p will be mentioned. We refer the reader to our supplementary document to see the figures and tables for those simulations. Then, we will summarize our findings from the empirical study.

In Model 1 (with a linear response), penalized linear models had a much lower MSE on test data than non-linear models, regardless of the values of n and p. This indicates that linear models are superior to non-linear models when it is appropriate to assume a linear relationship. On the other hand, linear models had a higher MSE on training data than

Table 3: Top five important genes chosen by lasso.

	Gene 1	**Gene 2**	**Gene 3**	**Gene 4**	**Gene 5**
Iteration 1	NBR2	DTL	VPS25	KHDRBS1	FLJ10241
Iteration 2	NBR2	DTL	C17orf53	VPS25	KHDRBS1
Iteration 3	NBR2	DTL	VPS25	CCDC56	TUBA1B
Iteration 4	NBR2	CCDC56	C17orf53	VPS25	DTL
Iteration 5	NBR2	DTL	CCDC56	TIMELESS	VPS25

Table 4: Top five important genes chosen by elastic-net.

	Gene 1	**Gene 2**	**Gene 3**	**Gene 4**	**Gene 5**
Iteration 1	NBR2	DTL	VPS25	C17orf53	CCDC56
Iteration 2	NBR2	DTL	C17orf53	VPS25	KHDRBS1
Iteration 3	NBR2	DTL	CCDC56	VPS25	KHDRBS1
Iteration 4	NBR2	CCDC56	C17orf53	VPS25	DTL
Iteration 5	NBR2	DTL	CCDC56	TIMELESS	TUBA1B

Table 5: Top five important genes chosen by MCP.

	Gene 1	**Gene 2**	**Gene 3**	**Gene 4**	**Gene 5**
Iteration 1	NBR2	DTL	VPS25	KHDRBS1	ANKRD13B
Iteration 2	NBR2	DTL	VPS25	KHDRBS1	ANKRD13B
Iteration 3	DTL	NBR2	VPS25	KHDRBS1	ANKRD13B
Iteration 4	NBR2	TUBA1B	RPL27	FLJ10241	ECH1
Iteration 5	NBR2	TIMELESS	RPL27	PGLYRP3	ANKRD13B

Table 6: Top five important genes chosen by random forests.

	Gene 1	**Gene 2**	**Gene 3**	**Gene 4**	**Gene 5**
Iteration 1	C17orf53	TUBG1	NBR2	DTL	VPS25
Iteration 2	TUBG1	NBR2	C17orf53	VPS25	DTL
Iteration 3	C17orf53	TUBG1	NBR2	DTL	VPS25
Iteration 4	C17orf53	NBR2	TUBG1	RDM1	TIMELESS
Iteration 5	TUBG1	NBR2	C17orf53	VPS25	MLX

non-linear models in almost all the simulations. We see that the training MSE for the linear models was very close to their test MSE. This indicates that the linear models did not have any overfitting, whereas the non-linear models did. In Model 2 (with a non-linear response), penalized linear models generally have a slightly higher test MSE when $p > n$ when compared to non-linear models. Still, in high-dimensional situations, the linear models are competitive with non-linear models even when the response cannot be assumed to depend linearly on the predictors. When $n > p$, the linear models had a significantly higher test MSE. This is expected, since a large number of predictors significantly lowers the variance of the non-linear

models. The linear models continue to have a high bias when n is large, which results in large test errors.

For both Model 1 and Model 2, SCAD and MCP generally had the lowest test MSE among the linear models. Lasso and elastic-net almost never performed as well as SCAD and MCP but were usually not significantly worse. Ordinary least squares and ridge regression were the worst-performing linear models. These two models likely suffered from their inability to perform variable selection. Because both Model 1 and Model 2 had only a small number of important predictors, the need for variable selection is pivotal for good model performance. Lasso and elastic-net tended to be pickier about the variables they selected compared to other linear models, especially when the correlation among predictors was high. This is evidenced by these models having a lower β-sensitivity and higher β-specificity. Even though SCAD and MCP were more likely to identify important predictors, all of the linear models struggled to identify important predictors in Model 2. This indicates that when the response cannot be assumed to have a linear dependence on the predictors, the linear models may not be reliable for inference.

The standard deviation of the random error did not have any broad qualitative effects on the results, nor did the correlation structure. The standard deviation of the random error did result in worse-performing models, but none of the models appeared to be affected more or less when varying the random error. Overall, we conclude that among the linear models, MCP generally had the best performance. This conclusion backs the results of similar simulation studies that compared penalized regression techniques [4, 43]. When p is large, the performance of MCP is comparable to that of the non-linear models.

Now, we will discuss some findings from the empirical analysis. We found that SCAD and MCP maintained the lowest testing mean squared error among the tested models. This can be seen in Figure 5 and Table 2. XGBoost, elastic-net, and lasso all had very close performances to SCAD and MCP. Ridge regression and random forest models performed the worst among the models considered. In addition to minimizing the test MSE, MCP also maintained the lowest standard deviation for the test MSE among the five cross-validation folds, as seen in Table 2. Lasso and SCAD also had small standard deviations. On the other hand, the non-linear models had very different performances on each fold, meaning that they are more sensitive to the training data used. These results are all expected, given that the linear models have high bias (resulting in low variance) while the non-linear models have low bias (and consequently high variance). The penalized regression models were fitted exceptionally faster than random forest and XGBoost as documented in Table 2. 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.

Recall that for lasso, elastic-net, MCP, random forests, and gradient boosts, we tracked the important predictors selected by each model. On all five cross-validation folds, all five models identified NBR2 as an important feature. This gene is known to be a neighbor of

the BRCA1 gene and acts as a tumor suppressant [41]. The genes DTL, TIMELESS, and KHDRBS1 (also known as Sam68), which each appeared in at least one fold, are known to be related to breast cancer [35, 18, 2]. The last gene, VPS25, is also known to be a tumor suppressant [37]. These results show the potential for these statistical models to reveal new insights about the roles of genes in areas such as oncology and demonstrate the selection accuracy of such methods.

From Tables 3 through 6, we see that there is some consistency between the top five genes identified by each model. In particular, all of the models tend to choose NBR2, DTL, and VPS25 as some of the most important genes. However, other genes only appear for some of the models tested. For example, C17orf53 is chosen as a top five predictor multiple times when using lasso, elastic-net, and random forests, but does not appear at all when using MCP. Some further investigation found that C17orf53 is chosen as the 10th or 11th most important predictor for MCP in all five cross-validation folds. A similar situation appears with the gene KHDRBS1, which appears multiple times with lasso, elastic-net, and MCP, but not for random forests. It turns out that KHDRBS1 is not among the top 15 genes for any cross-validation fold of random forests (it ranks 31, 16, 28, 26, and 16). These results show that the four models tested will generally identify the same set of genes as important, but the exact rankings chosen by each model may vary.

6. Conclusion. There is a severe lack of comprehensive testing comparing widespread machine learning methods such as random forest, gradient boosting, and support vector machines with more statistical methods such as 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. As complex "black box" machine learning and artificial intelligence models rise in popularity, the importance of statistical methods that perform variable selection, are easily understood, and simplify models becomes increasingly important.

These comparisons have shown that penalized regression should be added to the toolbox of any data scientist and machine learning engineer. Penalized regression models overall had better test performance than traditional machine learning models when applied to our linear simulation data and genomic data. For the non-linear simulation model, machine learning models attain a slightly better performance. We note, however, that penalized regression techniques offers several additional advantages over machine learning models. Specifically, they are typically much less computationally expensive compared to machine learning models. Ensemble algorithms, such as the random forest and gradient boosting models considered in this paper, can be especially slow. Also, in scenarios such as the empirical data study outlined earlier, penalized regression techniques can help identify the relationship between predictors and a response value. In such cases, the ability to determine these relationships can be more important than the predictive performance of a model. For genetic applications such as that of the empirical data studied, understanding important predictors allows for a streamline of the identification of genes that cause cancer and allows for scientists to specifically target important genes for cancer treatments.

In the future, it may be useful to develop and test 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 and would hopefully perform better than either random forest or penalized regression methods individually. Some such models already exist, as seen in [6, 9, 44] 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 variable selection using wrapper methods may result in an exponentially slower runtime. Thus, it is important that embedded variable selection methods, such as penalized regression methods, are utilized in ensemble methods.

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.

7. 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. We also thank Dr. Nicholas Luke, Dr. Karen Yokley, and Yang Xue for their guidance throughout the research process.

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

REFERENCES

[1] A. Antoniou, P. D. Pharoah, S. Narod, H. A. Risch, J. E. Eyfjord, J. L. Hopper, N. Loman, H. Olsson, O. Johannsson, Å. Borg, 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 (2003), pp. 1117–1130.

[2] P. Bielli, R. Busà, M. P. Paronetto, and C. Sette, *The rna-binding protein sam68 is a multifunctional player in human cancer*, Endocrine-related cancer, 18 (2011), pp. R91–R102.

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

[4] P. Breheny and J. Huang, *Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection*, The annals of applied statistics, 5 (2011), p. 232.

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

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

[7] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, M. Li, J. Xie, M. Lin, Y. Geng, and Y. Li, *xgboost: Extreme Gradient Boosting*, 2021, https://CRAN.R-project.org/package=xgboost. R package version 1.4.1.1.

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

[9] F. Degenhardt, S. Seifert, and S. Szymczak, *Evaluation of variable selection methods for random forests and omics data sets*, Briefings in bioinformatics, 20 (2019), pp. 492–503.

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

[11] C. Ding and H. Peng, *Minimum redundancy feature selection from microarray gene expression data*,

Journal of bioinformatics and computational biology, 3 (2005), pp. 185–205.
[12] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik, et al., *Support vector regression machines*, Advances in neural information processing systems, 9 (1997), pp. 155–161.
[13] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*, CRC press, 1994.
[14] J. Fan and R. Li, *Variable selection via nonconcave penalized likelihood and its oracle properties*, Journal of the American statistical Association, 96 (2001), pp. 1348–1360.
[15] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, Journal of statistical software, 33 (2010), p. 1.
[16] J. Friedman, T. Hastie, R. Tibshirani, et al., *The elements of statistical learning*, vol. 1, Springer series in statistics New York, 2001.
[17] J. H. Friedman, *Greedy function approximation: a gradient boosting machine*, Annals of statistics, (2001), pp. 1189–1232.
[18] A. Fu, D. Leaderer, T. Zheng, A. E. Hoffman, R. G. Stevens, and Y. Zhu, *Genetic and epigenetic associations of circadian gene timeless and breast cancer risk*, Molecular carcinogenesis, 51 (2012), pp. 923–929.
[19] W. H. Greene, *Econometric analysis*, Pearson Education India, 2003.
[20] I. Guyon and A. Elisseeff, *An introduction to variable and feature selection*, Journal of machine learning research, 3 (2003), pp. 1157–1182.
[21] A. E. Hoerl and R. W. Kennard, *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics, 12 (1970), pp. 55–67.
[22] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112, Springer, 2013.
[23] K. B. Kuchenbaecker, J. L. Hopper, D. R. Barnes, K.-A. Phillips, T. M. Mooij, M.-J. Roos-Blom, S. Jervis, F. E. Van Leeuwen, R. L. Milne, N. Andrieu, et al., *Risks of breast, ovarian, and contralateral breast cancer for brca1 and brca2 mutation carriers*, Jama, 317 (2017), pp. 2402–2416.
[24] Z. Li, T. Chen, Q. Wu, G. Xia, and D. Chi, *Application of penalized linear regression and ensemble methods for drought forecasting in northeast china*, Meteorology and Atmospheric Physics, 132 (2020), pp. 113–130.
[25] Z. Li and M. J. Sillanpää, *Overview of lasso-related penalized regression methods for quantitative trait mapping and genomic selection*, Theoretical and applied genetics, 125 (2012), pp. 419–435.
[26] A. Liaw and M. Wiener, *Classification and regression by randomforest*, R News, 2 (2002), pp. 18–22, https://CRAN.R-project.org/doc/Rnews/.
[27] X.-Y. Liu, S.-B. Wu, W.-Q. Zeng, Z.-J. Yuan, and H.-B. Xu, *Logsum+ l 2 penalized logistic regression model for biomarker selection and cancer classification*, Scientific Reports, 10 (2020), pp. 1–16.
[28] M. Lu, J. Zhou, C. Naylor, B. D. Kirkpatrick, R. Haque, W. A. Petri, and J. Z. Ma, *Application of penalized linear regression methods to the selection of environmental enteropathy biomarkers*, Biomarker research, 5 (2017), pp. 1–10.
[29] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2021, https://CRAN.R-project.org/package=e1071. R package version 1.7-7.
[30] D. Nielsen, *Tree boosting with xgboost-why does xgboost win "every" machine learning competition?*, (2016).
[31] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021, https://www.R-project.org/.
[32] N. Sánchez-Maroño, A. Alonso-Betanzos, and M. Tombilla-Sanromán, *Filter methods for feature selection–a comparative study*, in International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2007, pp. 178–187.
[33] R. E. Schapire, *The strength of weak learnability*, Machine learning, 5 (1990), pp. 197–227.
[34] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B (Methodological), 58 (1996), pp. 267–288.
[35] T. Ueki, T. Nishidate, J. Park, M. Lin, A. Shimo, K. Hirata, Y. Nakamura, and T. Katagiri, *Involvement of elevated expression of multiple cell-cycle regulator, dtl/ramp (denticleless/ra-regulated nuclear matrix associated protein), in the growth of breast cancer cells*, Oncogene, 27 (2008), pp. 5672–5683.

[36] M. G. Usai, M. E. Goddard, and B. J. Hayes, *Lasso with cross-validation for genomic selection*, Genetics research, 91 (2009), pp. 427–436.

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

[38] H. Wang, C. Liu, and L. Deng, *Enhanced prediction of hot spots at protein-protein interfaces using extreme gradient boosting*, Scientific reports, 8 (2018), pp. 1–13.

[39] L. Wang, G. Chen, and H. Li, *Group scad regression analysis for microarray time course gene expression data*, Bioinformatics, 23 (2007), pp. 1486–1494.

[40] M. N. Wright and A. Ziegler, *ranger: A fast implementation of random forests for high dimensional data in C++ and R*, Journal of Statistical Software, 77 (2017), pp. 1–17, https://doi.org/10.18637/jss.v077.i01.

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

[42] Y. Zeng and P. Breheny, *The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in r*, ArXiv e-prints, (2017), https://arxiv.org/abs/1701.05936, https://arxiv.org/abs/1701.05936.

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

[44] X. Zhang, Y. Wu, L. Wang, and R. Li, *Variable selection for support vector machines in moderately high dimensions*, Journal of the Royal Statistical Society. Series B, Statistical methodology, 78 (2016), p. 53.

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

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