Understanding and Estimating Predictive Performance of Statistical Learning Methods based on Data Properties

by

Haiyang Jiang

B.Sc., Simon Fraser University, 2017

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Statistics and Actuarial Science
Faculty of Science

© Haiyang Jiang 2020
SIMON FRASER UNIVERSITY
Summer 2020

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.

Approval

Name:	**Haiyang Jiang**
Degree:	**Master of Science (Statistics)**
Title:	**Understanding and Estimating Predictive Performance of Statistical Learning Methods based on Data Properties**
Examining Committee:	**Chair:** Richard Lockhart Professor
	Thomas Loughin Senior Supervisor Professor
	Lloyd Elliott Supervisor Assistant Professor
	Brad McNeney Internal Examiner Associate Professor
Date Defended:	**July 14, 2020**

Abstract

Many Statistical Learning (SL) regression methods have been developed over roughly the last two decades, but no one model has been found to be the best across all sets of data. It would be useful if guidance were available to help identify when each different method might be expected to provide more accurate or precise predictions than competitors. We speculate that certain measurable features of a data set might influence methods' potential ability to provide relatively accurate predictions. This thesis explores the potential to use measurable characteristics of a data set to estimate the prediction performance of different SL regression methods. We demonstrate this process on an existing set of 42 benchmark data sets. We measure a variety of properties on each data set that might be useful for differentiating between likely good- or poor-performing regression methods. Using cross-validation, we measure the actual relative prediction performance of 12 well-known regression methods, including both classical linear techniques and more modern flexible approaches. Finally, we combine the performance measures and the data set properties into a multivariate regression model to identify which properties appear to be most important and to estimate the expected prediction performance of each method.

Keywords: benchmarking; cross-validation; hyperparameter tuning; trees; splines

Acknowledgements

First and foremost, I owe my deepest gratitude to my supervisor Dr. Thomas Loughin. This thesis would not have been completed without his continued patience, support, and encouragement. His enthusiasm for statistics, tea, and many other topics made every weekly meeting pleasant and memorable. He is an excellent mentor who offered me so much wisdom in statistics and life that I could benefit from for years, and I feel fortunate to be one of his students.

I want to thank Dr. Richard Lockhart, Dr. Brad McNeney and Dr. Lloyd Elliott, for their time reviewing this project and being on my committee. Thank you to all the faculty members in the department. They brought me into the world of statistics and taught me to see the beauty of it. Also, I sincerely appreciate all the generous help from the staff members, Sadika Jungic, Charlene Bradbury and Kelly Jay.

My master's program's journey was filled with enjoyment, and I could not survive without that. I want to thank all of my colleagues from this and past cohorts for all the laughter we shared, and all the supports you gave. I especially want to thank Michael, Coco and Lucas for being through this program together and your kind help in the past two years, may our friendships last forever.

Also, I would like to take this space to send my gratitude to my colleagues and managers at the Royal Bank of Canada, who have helped me develop my professional career during the summer of 2019. Huge thanks to Song, Nassim, and Paula for being my team lead. Your kind mentorship shaped me to become a better data scientist. Thank you to Abdul, Parth and Leah, for being the best teammates that I could ever ask for. I am so inspired by your keenness and perseverance. I sincerely appreciate everyone at RBC who have helped and guided us to the project's accomplishment.

Finally yet importantly, I could not express enough gratitude to my loving wife, Vicky, and my family members. They always got my back and supported me unconditionally. I want to thank my wife for being with me through thick and thin, encouraging and cheering me every day. I am grateful to my in-law parents for being so understanding and kind during the tenure of my master's program. Thank you, my dog Dodo, for bring me so much pleasure and energy each day! Lastly, I want to thank my parents back in China, who selflessly encouraged me to study abroad and pursue the major I love. This journey would not have been possible if not for them, and I dedicate this milestone to them.

Table of Contents

Approval ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables vii

List of Figures viii

1 Introduction 1

2 Review of Statistical Learning for Regression 3

3 Review of Statistical Learning Methods 6

3.1 Methods based on Multiple Linear Regression . . . 6

3.2 Linear Regression with Variable Selection . . . 7

3.2.1 Variable Selection via Subsetting . . . 7

3.2.2 Variable Selection via Regularization . . . 8

3.3 Tree-based methods . . . 8

3.4 Methods based on Splines . . . 11

3.5 Artificial Neural Networks . . . 13

4 Review of previous comparisons made by others 14

5 Review of properties of data 17

5.1 Data Richness . . . 17

5.2 Multicollinearity . . . 18

5.3 Heteroscedasticity . . . 19

5.4 Signal Strength . . . 19

5.5 Non-linearity . . . 20

5.6 Interactivity . . . 20

5.7 Sparsity 21

6 Study Methodology **22**
6.1 Preparing Data 23
6.2 Obtaining predictive performance of SL methods 23
6.3 Modelling predictive performance based on data properties 25

7 Experimental Study Results **27**
7.1 Results of the measured data properties 27
7.2 Results of predictive performance 28
7.3 Multivariate regression model results 32

8 Discussion **36**
8.1 Discussion of results 36
8.2 Discussion of limitations 37

9 Conclusion and Future Works **41**

Bibliography **43**

Appendix A The table of pre-specified tuning parameter values for SL methods that require hyperparameter tuning using the `caret` package **45**

Appendix B Results from data property measurements on 42 benchmark data sets **47**

Appendix C Predictive performance from SL methods **51**

Appendix D Scatterplots of 7 measured data properties versus RelRMSPEs for the final 7 SL methods **55**

Appendix E Histograms of optimal hyperparameter values for 6 SL methods tuned using the `caret` package **62**

List of Tables

Table 6.1 A summary table of SL methods with the corresponding function names and packages in R. 24

Table 7.1 A summary table of estimated coefficients from the multivariate regression model with their standard errors. Entries in main body are coefficient (standard error). The last row contains the results of the Type II MANOVA Pillai tests in the form 'test statistic (p-value).' The coefficient estimates and test statistics that are statistically significant at $\alpha = 5\%$ level are shaded in grey. 35

List of Figures

Figure 7.1 Histograms of measured data properties on benchmark data sets after adjustments. 28

Figure 7.2 Correlation plot of RelRMSPEs for all 12 SL methods 29

Figure 7.3 Boxplot of RelRMSPEs from 1 to 2 for 7 SL methods. Number of RelRMSPEs larger than 2 for each method (that are not plotted) were the following: LASSO 7, Random forest 3, GBM 2, XGBoost 1, PPR 2, MARS 3, ANNs 2. 30

Figure 7.4 Boxplots of the RelRMSPEs for 41 benchmark data sets. 32

Figure 7.5 Scatterplots of the RelRMSPEs for each SL method versus nonlinearity. 33

Figure 7.6 Scatterplots of the RelRMSPEs for each SL method versus multicollinearity. 34

Figure 8.1 Scatterplots of the medians of RelRMSPEs across all 7 SL methods versus each data property. The blue straight lines represent the linear regression models. 40

Chapter 1

Introduction

As the idea of Machine Learning has become increasingly popular in recent years, more and more professionals in both academia and industry have begun leveraging Statistical Learning (SL) regression techniques to tackle data-related problems such as revenue projections and house price predictions. Many SL methods have been developed over roughly the last two decades, but no one model has been found to be the best across all sets of data. Numerous studies have compared various SL methods for regression problems (e.g., [5][12][19]), and found that different SL methods could perform relatively well or poorly in different data sets.

It would be useful if guidance were available to help identify when each different method might be expected to provide more accurate or precise predictions than competitors. However, we are not aware of any such guidance. This leaves a data analyst in a quandary when it comes to deciding which SL regression method to use on a given data set. One option is to train many candidate models and find out which one optimizes some pre-specified metrics. While this might be feasible for small data sets, it is not a general solution that could be applied in every situation. In particular, the computational time may become prohibitive, especially when training SL methods that require hyperparameter tuning using resampling techniques such as cross-validation or bootstrapping. It would be helpful to find an approach that can estimate an SL method's potential prediction performance reasonably accurately for any given set of data.

We speculate that certain measurable features of a data set might influence methods' potential ability to provide relatively accurate predictions. For instance, the linear regression model would be optimal for data with a linear mean trend, but might perform poorly on data with a substantial non-linear trend. Thus, the degree of non-linearity of a data set could potentially be used to predict the relative ability of linear regression methods to make accurate predictions. Gelfand [12] showed that the presence of heteroscedasticity could affect the relative prediction performance of a variety of SL methods. Other properties, such as sample size, number of explanatory variables, signal strength, and so forth, might also relate to when one SL regression method might give more accurate predictions than another.

This thesis explores the potential to use measurable characteristics of a data set to estimate the prediction performance of different SL regression methods. We demonstrate this process on an existing set of 42 benchmark data sets [5]. We measure a variety of properties on each data set that might be useful for differentiating between likely good- or poor-performing regression methods. Using cross-validation, we measure the actual relative prediction performance of 12 well-known regression methods, including both classical linear techniques and more modern flexible approaches. Finally, we combine the performance measures and the data set properties into a multivariate regression model to identify which properties appear to be most important and to estimate the expected prediction performance of each method. The short-term goal of this project is to prove the concept that properties of data sets can estimate the prediction performance for different SL methods. The long term goal is to provide a tool that could enable practitioners to foresee how a variety of candidate SL methods might work on their own data and to provide researchers a way to test and understand the relative performance of new SL regression methods. We also hope that this work can spur further research on understanding when different SL methods may work well.

The outline of this report is as follows. In Chapter 2, we review some important properties of SL for regression, and review some essential techniques to provide reliable estimation of the genenralization error. In Chapter 3, we review some popular SL candidates for regression problems. In Chapter 4, we review some comparisons between different SL methods previously made by other authors. In Chapter 5, we identify seven properties of data that we speculate may help to determine when different SL regression methods may predict well or poorly, and we propose ways to measure the properties. We introduce the detailed methodology of our study in Chapter 6, and we highlight some experimental study results and discussions in Chapters 7 and 8. We finally conclude this thesis and discuss ideas for future progress toward the long-term goal in Chapter 9.

Chapter 2

Review of Statistical Learning for Regression

In this thesis, we focus on Statistical Learning (SL) techniques for regression. By this, we mean methods or algorithms that take a numeric response and explanatory variables of any type from an observed data set as inputs and return predicted values on future data as outputs.

We first define some common notation needed throughout the thesis. We let Y be generically a response random variable, and $\boldsymbol{Y}_{n\times 1}$ be the vector of observed responses from a sample of size n, with elements y_i, $i = 1, \ldots, n$. These n observations are drawn independently from the random variable Y. Let X be the p-dimensional random variable representing the explanatory variables $X_1, \ldots, X_p$. The sampled data are represented by the matrix $\boldsymbol{X}_{n\times(p+1)}$, consisting of a column of ones followed by p columns representing observed values for individual variables. Let x_i represent one observation of all p variables, and let x_{ij} represent the i-th observation of the j-th variable, $i = 1, \ldots, n$; $j = 1, \ldots, p$.

A typical assumption in regression is that

$$Y = f(X) + \epsilon \tag{2.1}$$

where $f(\cdot) = E(Y|X)$ is an unknown real-valued function that describes the relationship between X and the mean of Y, and ϵ represents the uncontrollable random error with $E(\epsilon) = 0$ and constant variance, $\text{Var}(\epsilon) = \sigma_\epsilon^2$. The goal of SL for regression problems is to approximate the unknown $f(\cdot)$ with some function $\hat{f}(\cdot)$ based on some data in a training set $\mathcal{T} = (\boldsymbol{X}, \boldsymbol{Y})$. Different SL methods place different explicit or implicit structural constraints on the family of functions from which $\hat{f}(\cdot)$ may be chosen. The goal, then, reduces to selecting a member of the family such that its prediction for any future x_0 drawn from X, $\hat{y}_0 = \hat{f}(x_0)$, is close to $y_0 = f(x_0) + \epsilon$. Since ϵ cannot be predicted, the task amounts to developing a prediction model for $f(X)$.

To narrow down the choice of $\hat{f}(\cdot)$, we need a function to define 'close.' A loss function $L(Y, f(X))$ measures how far off the predictions are from the true mean. The most popular choice for regression is the squared-error loss

$$L(Y, f(X)) = (Y - f(X))^2, \tag{2.2}$$

due to its combined mathematical and computational convenience [14]. The goal is to find $\hat{f}(\cdot)$ that minimizes the generalization error or expected prediction error (EPE)

$$\text{EPE}(f) = E\left((Y - f(X))^2\right) = \text{E}_X \text{E}_{Y|X}\left([Y - f(X)]^2 | X\right) \tag{2.3}$$

among all members of the family [14]. Unfortunately, EPE in (2.3) is a theoretical quantity that we cannot measure directly. Using the sample mean to estimate the expectation, one estimate of (2.3) is the sample mean-squared error (sMSE) with the form

$$\text{sMSE}(Y, \hat{f}(X)) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}\left(x_i\right)\right)^2. \tag{2.4}$$

based on the sampled data (X, Y).

However, when $\hat{f}(\cdot)$ fits Y too well, it absorbs some of the irreducible variability into its predictions, which leads to inflated prediction errors. We refer to this scenario as 'overfitting'. It should be apparent that models that overfit will result in sMSE values that underestimate generalization error, especially when the complexity of $\hat{f}(\cdot)$ increases. In fact, with many SL methods, it is possible to perfectly fit the sample data and drive sMSE to zero. Therefore, a new (test) data set is needed, separate from $\mathcal{T}$, on which to estimate the population expected loss. We then use the training data set $\mathcal{T}$ for training $\hat{f}(X)$ and a test data set $\mathcal{T}^* = (\boldsymbol{X}^*, \boldsymbol{Y}^*)$ with n^* observations for estimating the conditional expectation in (2.3). The root mean-squared prediction error (RMSPE)

$$\text{RMSPE}(Y^*, \hat{f}(X^*)) = \sqrt{\frac{1}{n^*} \sum_{i=1}^{n^*} \left(y_i^* - \hat{f}\left(x_i^*\right)\right)^2} \tag{2.5}$$

is an unbiased estimate of EPE.

Data splitting and resampling techniques, such as K-fold cross-validation and bootstrap, are helpful in producing test data sets. A single data split partitions a given data set into training and test sets. However, the size of each set is only a fraction of the original size, which increases the variability of both the estimated prediction function and RMSPE. Repeated data splitting reduces the variability of the estimate of generalization error. The K-fold cross-validation (CV) procedure is a popular method for performing multiple splits in a systematic way. In CV, the training data are partitioned into K subsets called 'folds.'

It sequentially sets the k-th fold to be the test (validation) set, and the remaining to be the training set, for $k = 1, 2, \ldots, K$.

Analysis of EPE in (2.3) reveals that it can be split into three components: irreducible error, the squared bias of the estimator, and the variance of the estimator [14]. The irreducible error is inherent variability that we cannot control. Thus, minimizing (2.3) requires minimizing the sum of the squared bias and variance. Unfortunately, this cannot be achieved componentwise, because there is an inherent trade-off between bias and variance. Functions with high flexibility (high model complexity) can fit closer to each data point in the training set to reduce bias. However, more parameters are required to achieve this flexibility, and estimating these parameters increases variability. In contrast, if a model is overly simple, it may have low variability but fail to adequately capture the true structure of $f(\cdot)$, resulting in high bias. A balance between bias and variance needs to be found to minimize the generalization error.

Different SL methods control the balance between bias and variance differently. Some assume specific structures for the mean (e.g. multiple linear regression), and their training processes consist of estimating the model parameter values to minimize loss. Many other methods allow forms for $\hat{f}(\cdot)$ that adapt to the data and use algorithms to generate the estimates. These methods often involve tuning 'hyperparameters' that control how closely the algorithm fits the data. A hyperparameter (equivalently, a tuning parameter) is a parameter that controls some aspect of an estimation algorithm and whose value must be set before the learning process begins [6].

Hyperparameters are often numeric values that lie within some practical limits, but optimal values for a given problem are usually not known. To estimate (or 'tune') them, some search algorithms can be used, but more commonly they are estimated based on minimizing estimates of the generalization error of the algorithm under different candidate values. Generalization errors in this case are typically estimated by splitting the training data into training and 'validation' sets, including using techniques such as CV. Where some data splitting technique is already being used to create test sets for estimating final generalization error of a SL method, validation sets must be created *after* this splitting. In this way, the RMSPE computed on the test set reflects the uncertainty from estimating the SL method's hyperparameters.

Chapter 3

Review of Statistical Learning Methods

For our study, we investigated four types of often-used SL methods: methods based on multiple linear regression, tree-based methods, methods based on splines, and artificial neural networks (ANNs). In this chapter, we review each method.

3.1 Methods based on Multiple Linear Regression

As the most fundamental regression technique, multiple linear regression (MLR) often serves as a quick tool in practice to build an SL model. It is fast and easy to interpret. Multiple linear regression fits the response variable with a linear hyperplane. Specifically, it has the form,

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i, \tag{3.1}$$

which is equivalent under the matrix form [14] to

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.2}$$

where $\boldsymbol{\beta}$ is a $(p+1)\times 1$ coefficient vector with elements $[\beta_0, \beta_1, \ldots, \beta_p]'$, and $\boldsymbol{\epsilon}$ is a $n\times 1$ vector for random errors that follow $\mathcal{N}(\mathbf{0}_n, \sigma_\epsilon^2 \boldsymbol{I}_n)$. The ordinary least squares (OLS) estimate of $\boldsymbol{\beta}$ minimizes

$$\mathrm{RSS}(\boldsymbol{\beta}) = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}), \tag{3.3}$$

which is the matrix expression of (2.2) under model (3.2), and it results in

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X'X})^{-1}\boldsymbol{X'Y}. \tag{3.4}$$

Multiple linear regression's rigid structure makes it produce stable but possibly biased predictions [14]. It can be made less rigid by incorporating a variety of variable types, including quantitative variables, transformations of quantitative variables (e.g. log, square-

root, etc.), polynomial terms (e.g. $X_2 = X_1^2$, $X_3 = X_1^5$, etc.), interaction terms (e.g. $X_3 = X_1 \times X_2$), and discrete variables using indicators for levels. However these terms must be specified manually, and MLR assumes that all necessary variables are specified. Failing to include an important variable adds bias, while estimating parameters for unnecessary variables adds variance. Whether to add or remove a particular variable is a tradeoff between the relative sizes of these two errors.

3.2 Linear Regression with Variable Selection

In many problems it is not fully known in advance which variables in a data set are useful for making predictions, especially in consideration of the bias-variance tradeoff. Variable selection techniques allow the data to adaptively identify which variables improve generalization error; thus they can be very useful in managing the tradeoff. In this thesis, we implemented two types of variable selection schemes: subsetting the variable space, and setting some coefficients to zero through shrinkage methods.

3.2.1 Variable Selection via Subsetting

The idea of subsetting is to find a subset of variables from the original set of variables that optimizes some selection criterion. There are two classic approaches for subsetting: stepwise regression and all-subsets regression. Common selection criteria include the residual sum of squares (RSS) and information criteria (IC).

All-subsets regression considers all possible variable subsets of the original variable space. For a data set with p variables, all-subsets regression fits all 2^p models and finds the models that minimize RSS for each model size $k \in \{0, 1, 2, \ldots, p\}$. Then it compares models of different sizes based on IC or some other criterion. The choice of k is a bias-variance tradeoff because a larger k induces less bias and more variance. All-subsets regression is not practical when p is too large. For example, the `leaps` package in R can handle a data set with at most 49 variables.

Stepwise regression finds the subset of variables differently. It creates a sequence of models to consider as candidates and then selects the one that optimizes some criterion (e.g. IC). There are numerous algorithms for creating the sequence of candidates and performing the selection. The one we implemented is based on the `step` function in R. It builds the path by starting with a null model (i.e. a linear model with only the intercept) and adds one variable that reduces RSS the most. Then in each step, it either adds the one variable that reduces RSS most; or drops the one that raises RSS the least. The decision is made according to which step lowers the IC from the previous step the most. The path-building stops either when neither adding nor dropping variables can lower the IC from the previous step.

3.2.2 Variable Selection via Regularization

Instead of going through discrete sets of candidate models, linear regression models with regularization focus on penalizing the loss function during estimation to encourage variable selection. For regularization methods we implemented here, the parameter estimates are no longer unbiased, but they often achieve lower variance, and hence lower prediction error.

One popular penalized regression technique is the Least Absolute Shrinkage and Selection Operator (LASSO) [28]. The idea of LASSO is to add a L1-norm penalty on the size of the coefficients to the squared-error loss function, and find $\boldsymbol{\beta}$ that minimizes the resulting objective function,

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\,. \tag{3.5}$$

The tuning (regularization) parameter λ balances the optimization between minimizing the RSS and keeping the parameter vector small. If $\lambda = 0$, the parameter estimation of LASSO is the same as for multiple linear regression. As λ increases, $\sum_{j=1}^{p}|\beta_j|$ must shrink to minimize (3.5), and it results in zero coefficients for less useful variables due to L1 norm's geometric property [14]. Thus LASSO serves as both a shrinkage method and a variable selection scheme.

The optimal value of λ is a bias variance trade-off. The larger the λ, the more sparse the solution tends to be, which leads to lower variance but higher bias. The choice of λ is usually the one that minimizes CV error. An alternative way to choose λ is the 'one-standard-error' ('1-SE') rule proposed by Hastie et al. [15], where we choose the largest value of λ with a CV-error no more than one standard error above its minimum value.

One variant of LASSO we included here is the Relaxed LASSO, which can lead to lower prediction errors compared to LASSO [14]. It first runs LASSO with different values of λ to select sets of variables without estimating their coefficients. Then it reruns LASSO on the selected variables only, but with a penalty $\phi\lambda$ instead of λ, where $\phi \in [0, 1]$ [14]. The hyperparameter ϕ is considered as a 'relaxation parameter' of the $\boldsymbol{\beta}$ coefficients. When $\phi = 0$, there is no penalty on $\boldsymbol{\beta}$, and the parameter estimates become the OLS estimates of the variables selected in the first iteration. When $\phi = 1$, the parameter estimates are the same as the original LASSO parameter estimates. Setting ϕ between 0 and 1 results in a mixture of OLS and LASSO estimates. Both ϕ and λ are tuning parameters. The optimal value of ϕ depends on the value of λ, so these two hyperparameters need to be chosen jointly.

3.3 Tree-based methods

Linear regression models work in many cases but are of limited use when the response surface is highly nonlinear or contains interactions that were not specified in the model.

Tree-based methods can simultaneously adapt their shapes to arbitrary response surfaces and incorporate unforeseen interaction effects into the model. They approximate $f(\cdot)$ with a piecewise constant surface, where the constants are mean response values within regions of X. The regions are determined adaptively using a recursive partitioning algorithm.

Specifically, the algorithm first considers splitting data into two regions (or 'nodes') R_1 and R_2 of the form

$$R_1 = \{i : x_{ij} \leq c\}$$
$$R_2 = \{i : x_{ij} > c\},$$

for selected values of c and all explanatory variables $j \in \{1, \ldots, p\}$, where c represents the point to split on for each variable. For each j and c, the sample means $\bar{y}_1$ in R_1 and $\bar{y}_2$ in R_2 are computed. An optimal split is chosen to maximize the reduction in RSS after the split compared to before. For the first split, the RSS for the full data before the split is

$$RSS(\text{Full Data}) = \sum_{i=1}^{n} (y_i - \bar{y})^2;$$

and the new RSS after the split becomes

$$RSS(\text{Split}) = \sum_{i \in R_1} (y_i - \bar{y}_1)^2 + \sum_{i \in R_2} (y_i - \bar{y}_2)^2.$$

Once the first split is created, each of the regions created previously is split into two new subregions by the recursive application of the same process. Each new split may be on a different variable or on the same variable at a different level. The regression tree keeps splitting recursively until some stopping criterion is reached. There are two common stopping criteria. One is to stop if further reduction in RSS from a subsequent split is below some specified threshold; the other is not to consider any splits that place fewer than C_1 observations in a subregion, for a pre-specified value C_1. The prediction function then consists of the sample mean computed within each of the final subregions ('terminal nodes') that is not split further.

Trees may overfit or underfit depending on the sample size and the stopping rule. When the tree overfits, it may worth combining subregions into larger regions to smooth the mean more, thus lowering the RMSPE. The process of combining subregions is called 'pruning.' Trees are pruned using the 'cost-complexity' criterion (more details in [3]), which balances reduction in RSS against increasing model complexity. The amount of pruning to be applied to a given tree is determined using CV. The optimal size of the tree is the one that minimizes the CV error, or the smallest tree that has its estimated error within one standard error of the smallest estimated CV error (the '1-SE' rule).

Despite the popularity of its structure, a single regression tree is usually not a competitive predictor because the surfaces it produces tend to be excessively variable. One small change in the data could lead to creation of entirely different regions due to the recur-

sive partitioning. Also, the piecewise constant model may be too coarse when $f(\cdot)$ changes rapidly unless there are many data points in the regions of rapid change. One way to improve predictions from a tree is the ensemble method, whose main idea is to combine multiple base learners (single regression trees in this case) in a certain way such that the final predictions are less variable than the individual components [14]. Two common ensemble techniques are bagging and boosting.

Bagging builds multiple trees, each with bootstrapped samples, and the resulting predictions are averaged across trees. To see how this helps, consider the following simplified scenario. Suppose that prediction errors from each of B trees at a given x are identically distributed with variance σ^2 and pairwise correlation ρ, then the variance of the average of these trees is $\sigma^2(\rho + (1-\rho)/B)$. Assuming that bootstrapping does not add significant variance to the trees' prediction errors, then the ensemble has smaller variance than a single tree. Increasing B reduces the variance, but the potential reduction is limited if the correlation among predictions errors from different trees is large.

Random Forest (RF) implements bagging with one change. At each split, only $m \leq p$ randomly sampled explanatory variables are considered for splitting. Choosing $m < p$ may increase the variance of individual trees by forcing tree splits to sometimes take place suboptimally. However, the resulting trees produce prediction errors that are less correlated and hence the ensemble can have lower overall variance than with $m = p$. The tuning parameter m is chosen to minimize a built-in measure of validation error.

Unlike bagging or RF, boosting methods 'boost' the process of learning a mean surface by sequentially piecing together a large number of functions that learn only a small part of the mean surface. Boosted regression trees first fit observations with the overall mean response. Then in each iteration, the algorithm fits a small regression tree to the residuals from the previous fit, and uses the fitted values from the new tree to augment predicted values. These steps are repeated M times, and M is a tuning parameter. Another parameter ν controls how much each new tree influences the predicted values and encourages slow learning. The size of each tree J (which is another tuning parameter) is kept relatively small to encourage slow, thorough learning of the entire surface [17]. Stochastic Gradient Boosting proposed by Friedman [10] uses only a fraction of the full data set for training each tree in the sequence, which can improve the performance of a boosting machine. The fraction of the data chosen at each step, η, is another tuning parameter. All these tuning parameters are tuned through CV.

There are several implementations of boosted regression tree algorithms with different features. We implement two of these among our SL methods. Gradient Boosting Machine (GBM) is a popular implementation that connects the 'residual' in boosted regression trees with the gradient of the loss function, so that GBM serves as a steepest descent algorithm. Extreme gradient boosting (XGBoost) prevents overfitting in GBM by using a form of

regularization in the tree learning phase. It also applies parallel computing in a novel way to speed up the training process. More details are covered in [4].

3.4 Methods based on Splines

Methods based on splines fit surfaces that are more flexible than a linear hyperplane, but smoother than what trees produce. The core idea of regression (basis) splines is to partition the domain of X into subregions, and fit polynomial regression models (usually cubic) on them, with constraints to maintain smoothness on the entire function. Assuming that X is one-dimensional ($p = 1$), the partitioning then occurs on the real line. The join points between partitioned intervals are called knots, and the number of knots is a tuning parameter. Increasing the number of knots leads to a more flexible fit.

An alternative spline technique is called smoothing splines, which uses an unlimited set of basis functions but regularizes the coefficients to control the fit. Again assuming $p = 1$, smoothing splines try to find the $f(\cdot)$ that fits the data best subject to the squared-error loss with a penalty on the second derivative of $f(\cdot)$ to promote smoothness. The penalty coefficient is a tuning parameter. The solution turns out to be a natural cubic spline with knots at each unique value of X and parameters estimated with regularization to prevent overfitting. In practice, it usually suffices to place $k < n$ number of knots (i.e. basis dimension) for some k, where k is a tuning parameter.

Although regression and smoothing splines are most easily defined when $p = 1$, most SL regression problems have $p > 1$. There exist multivariate versions of splines, but fitting them with good precision requires much larger samples as p increases. A more practical approach is the Generalized Additive Model (GAM), which is similar to a MLR model with a smoothing spline in each dimension. A GAM model under the regression setting has the form

$$Y = \alpha + \sum_{j=1}^{p} f_j\left(X_j\right) + \epsilon,$$

where $f_j(X_j)$'s are smoothing splines, and the error term ϵ has the mean zero. The splines in a GAM are fit to each dimension adaptively, using a special algorithm that cycles through dimensions iteratively [16].

Several popular algorithms have been developed based on the ideas of splines. One is Multivariate Adaptive Regression Splines (MARS). MARS combines the strength of both trees and splines, having the ability to perform variable selection and automatic interaction detection, while adapting to the data in a continuous manner. MARS models consist of pairs of piecewise linear basis functions of the form

$$h_{m1} = \begin{cases} x - t_q, & \text{if } x > t_q, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad h_{m2} = \begin{cases} t_q - x, & \text{if } x < t_q, \\ 0 & \text{otherwise,} \end{cases}$$

where the knots, t_q, are chosen from among the values x_{ij} for functions based on X_j [14]. These two functions are called a 'reflective pair' of hinge functions due to their shapes. MARS estimates Y with a linear combination of M basis functions [14],

$$\hat{Y} = \sum_{m=1}^{M} \sum_{q=1}^{2} \beta_m h_{mq}(X),$$

where the basis functions in the model are chosen from a prescribed 'library' of reflective pairs in a forward stepwise manner as follows.

MARS first takes the mean of Y as the predictions. All possible reflective pairs for the given data $\boldsymbol{X}$ constitute an initial library of basis functions for the first step. Then at each subsequent step, the library is augmented by adding all products of a candidate pair in the library with a basis function in the model, up to the interaction degree d chosen for the model (d is a tuning parameter). Similar to forward stepwise regression, the choice of $h_{mq}(X)$ at each step is based on minimizing RSS. The forward stepwise regression continues until some stopping rule (e.g. maximum number of terms M, a tuning parameter) is reached. The full model typically overfits. Thus backward elimination of individual hinge functions is then used to select the smaller model (usually) that produces the lowest generalized cross-validation (GCV) error, with details in [16].

The second popular algorithm related to GAM is Projection Pursuit Regression (PPR). Whereas GAM fits a spline function in each dimension and makes a linear combination of them, PPR reverses this process. We define $\boldsymbol{X}$ here as the $n \times p$ data matrix. PPR first makes a linear combination of all X_j's by using a $p \times 1$ unit vector ω, which represents a direction in p-dimensional X-space, and $\boldsymbol{X}\omega$ represents a projection of the n data points onto that direction. The goal of selecting ω is to find the direction in which Y changes the most. Then PPR fits an arbitrary univariate function $g(\cdot)$ (e.g. a smoothing spline) to the plot of Y against $\boldsymbol{X}\omega$ and uses it to predict Y in the given direction. The above process is iterated M times using the residuals from the previous fits, resulting in M functions $g_1(\cdot), \ldots, g_M(\cdot)$, and M unit vectors $\omega_1, \ldots, \omega_M$. Finally, PPR combines all functions into one prediction as an additive model with parameters estimated by least squares,

$$f(\boldsymbol{X}) = \sum_{m=1}^{M} \beta_m g_m \left(\boldsymbol{X}\omega_m\right).$$

More details about the fitting process are in [11]. Again, the number of iterations M is a tuning parameter that can be chosen by cross-validation.

3.5 Artificial Neural Networks

Artificial Neural Networks (ANNs) use a similar idea as PPR but with different specifications. There are many different forms of ANNs, many of which have shown excellent prediction performance. We focus here on single-layer neural network models to make studying them feasible.

A single-layer ANN model first creates M 'hidden nodes', $Z_1, \ldots Z_M$, from inputs $X_1, \ldots, X_p$ such that $Z_m = \sigma\left(\mathbf{X}\boldsymbol{\alpha}_m\right), m = 1, \ldots, M$. The vector $\boldsymbol{\alpha}_m = [\alpha_{0m}, \alpha_{1m}, \ldots, \alpha_{pm}]'$ contains 'weights' on the data matrix. The 'activation function' σ is the same for all m, and it is often the sigmoid function $\sigma(v) = 1/\left(1 + e^{-v}\right)$. For regression problems, an ANN then creates a linear combination of all hidden nodes to make predictions of Y: $\hat{Y} = \boldsymbol{Z}\boldsymbol{\beta}$, where $\boldsymbol{Z} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m]$, and $\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots, \beta_M]'$ is another set of weights. A single-layer neural network model estimates the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by using an algorithm called back-propagation; details are in [14].

An ANN can be a heavily parameterized function when M is large. It requires estimating $M(p+2)+1$ parameters. Thus, regularization is needed to control the model complexity. Define $||\boldsymbol{x}|| = \sum_{i=1}^{n} x_i^2$ for any $n \times 1$ vector $\boldsymbol{x}$. Instead of minimizing RSS, ANNs minimize RSS$+\lambda\{||\boldsymbol{\beta}||^2 + |||\boldsymbol{\alpha}||^2\}$ during the parameter estimation process. The penalty term λ is called the 'decay', which is a tuning parameter. The number of hidden nodes M is another tuning parameter, and they can both be chosen through CV.

Chapter 4

Review of previous comparisons made by others

It is common that researchers gauge a newly-developed SL method's performance by comparing it with other SL methods that perform the same task. Comparisons tend to be based mainly on simulated data sets, where properties of the data can be carefully controlled.

For instance, if a new SL method claims to capture a non-linear trend better than previous methods, one could simulate data sets with various levels of non-linearity, and compare the performance of the new SL method with existing ones. Replicating simulated data sets allows one to more accurately measure and compare prediction performance of different SL methods. However, since simulation data sets use simplified models of real data-generating mechanisms, comparisons made in simulations may not reflect actual performance potential in real data.

Comparisons of SL methods on real data have the potential to provide a clearer understanding of methods' relative predictive performance in realistic settings. However, the lack of control over properties of the data-generating mechanism inhibits understanding of why some methods perform better than others. Furthermore, there seems to be no systematic approach to selecting data sets for this task. Even though many, many papers have been published where methods are used on real data, the number of data sets used for each comparison tends to be small, and the reason for choosing those data sets is sometimes unclear. For example, those data sets might be handpicked to highlight the strengths of the newly-developed method, while failing to demonstrate its potential weaknesses. Therefore, even though one SL method may perform well on a small number of data sets, it may fail when extrapolated to a broader range of problems [23]. Thus, having a pre-chosen suite of 'benchmark' data sets for comparisons is useful. It removes potential for bias in the selection of data sets and ensures that SL methods are then compared under data sets with a variety of different properties, although the properties are not well controlled.

At present there is no consensus on which data sets to use in a benchmark suite, nor on what properties those sets should possess. However, some progress is being made. Olson

et al. developed the Penn Machine Learning Benchmark (PMLB), a publicly available data set suite with both real-world and simulated data sets [23]. They initially focused on supervised classification problems[1], and introduced a list of 'meta-features' that characterize data sets in many aspects, such as the number of observations and the number of continuous variables. After evaluating 13 popular classification SL methods, they used clustering methods to study which meta-features could lead to useful predictions, which data sets appear to be particularly useful for highlighting differential SL algorithm performance, and other comparisons. However, their clustering analysis had no capacity for predicting which SL methods might be better under which circumstances. Also, the properties that they measured were limited to measurements about the size and shape of the data matrix. They made no effort to measure internal properties of the data such as collinearity and signal strength.

Chipman et al. used a different suite of benchmark data sets in the context of demonstrating performance of their Bayesian additive regression trees (BART) algorithm [5]. They used an interesting approach to make predictive performance comparisons of BART with competing methods on 42 real-world data sets [5][2]. Specifically, they used multiple train/test splits for each data set, trained and tested all SL methods on each split, computed the relative RMSPE (RelRMSPE) on each split, and summarized results across data sets. They defined RelRMSPE as the RMSPE divided by the minimum RMSPE obtained by any SL method for each split. Thus, a method obtained an RelRMSPE of 1 when that method had the lowest RMSPE on that split. As opposed to the RMSPE, the RelRMSPE provides meaningful comparisons across data sets because of its invariance to the scale of the response variables [5]. They presented the results across all 42 data sets simultaneously, demonstrating which methods tended to perform best or nearly the best most often, which ones were consistently good or bad, and which ones might have had very variable performance. However, they lacked a way to connect the differential performance of the methods with the different characteristics of the data sets. Thus, they cannot understand the circumstances that might lead to one method's performing relatively well or poorly.

We suspect that the relative performance of different SL methods in real data sets, as measured by Chipman et al. [5], may be related to properties of those data sets. This is known to some extent. For example, linearity, sample size, and number of variables are all known to affect performance of different methods [17]. We propose to numerically measure properties like non-linearity, heteroscedasticity, collinearity, and so forth, on a variety of data

[1]The authors gathered an additional 120 data sets for regression problems after their paper was published. These data sets are hosted on the same site as the ones for classification problems, which is https://github.com/EpistasisLab/penn-ml-benchmarks. However, they do not appear to have performed any formal analysis on these data sets yet.

[2]We excluded 3 simulation data sets they applied. And the original sources of these data sets are from Kim et al. [19].

sets in a benchmark suite. We can then model the relative predictive performance of each SL method by regressing their RMSPEs on the measured properties, and determine which methods are sensitive to which data properties. Finally, the model can help us predict which SL methods are likely to perform well on any new data set. Surprisingly, to our knowledge, there is no previous study on predicting an SL method's performance based on measurable properties of the input. While we demonstrate this on a limited group of data sets, SL methods, and data properties, our ideas are flexible and can easily be extended in each dimension to achieve wider generalizability and less variability. Anyone with a new data set can easily measure its properties, and estimate the predictive performance of various SL methods. And researchers developing new SL methods can test them out in an 'arena' built to facilitate understanding of how the new methods work.

Chapter 5

Review of properties of data

To demonstrate how our proposed arena can work, we identified and measured 7 properties of data sets that could potentially affect the predictive performance of candidate SL methods. The properties, which are described more rigorously later in this chapter, are data richness, multicollinearity, heteroscedasticity, signal strength, non-linearity, interactivity and sparsity. We summarize each property into one numerical value per data set on a standardized scale, so that values computed on different data sets are directly comparable, and we use them later as inputs to model SL methods' relative performance.

Some of the 7 data properties we investigated are well-known to affect the relative performance of different methods [17], some have evidence indicating possible effects (e.g. the effects of heteroscedasticity presented in [12] and [26]), and some we speculate are potentially important. However, for several of these properties, there is no established numeric measure. In these cases, we seek measurements that are easy to implement, reflect the corresponding data property, and avoid contamination from other properties. Many of our proposed measures are variants of F-statistics comparing RSS from different models that attempt to isolate the properties of interest. Such F-like statistics are unitless and can be readily compared across response variables with different scales.

We acknowledge that the properties proposed here are not the only ones that could affect SL methods' predictive performance, and the measurements we use might not be perfect. Our goal, for now, is to develop 'quick and dirty' measures that are sensitive and specific to the features we measured. More thorough testing, analysis, and development of data-property measures is reserved for future work.

5.1 Data Richness

We define data richness as the amount of data compared to the dimension of X, representing potential complexity of an SL model. Its importance is related to the curse of dimensionality. 'Sampling density', or points per unit area, reflects how much data there is locally for estimating flexible SL methods. The more dense the data are, the more complex models

can be fit, potentially reducing bias without vastly increasing variance. With lower density, complex models become excessively variable, and one has to reduce complexity, possibly admitting more bias.

One can show that the sampling density, or points per unit area, is proportional to $n^{1/p}$ [14]. We adopt the measure of sampling density from [14] as our measure of data richness, i.e., $n^{1/p}$. As a part of the data pre-processing, we convert a categorical variable with K levels into $(K-1)$ dummy variables. Thus, we measure the value of p by summing the number of continuous variables and the number of dummy variables, for a data set with categorical variables.

5.2 Multicollinearity

Multicollinearity is defined as the existence of near (or complete) linear dependency among column vectors of the design matrix $\boldsymbol{X}$ [24]. Recall that the calculation of (3.4) requires that the inverse of $\boldsymbol{X'X}$ exists, thus $\boldsymbol{X}$ must be of full column-rank. When columns of $\boldsymbol{X}$ are perfectly collinear, the inverse does not exist. When they are nearly collinear, the inverse is unstable and sensitive to small changes in $\boldsymbol{X}$, which results in extreme variability in $\hat{\beta}$. Multicollinearity is not problematic as long as all related variables are in the true model and the prediction of new data does not require extrapolation, but it is a problem when the collinear variables are not all required in the model, especially when extrapolation happens. More generally, multicollinearity can add noise into regressions by interfering with the ability to identify and estimate the effects of the correct variables. Studies have shown that for projection-based methods such as ANN and PPR, multicollinearity in the design matrix affects the dimension of the projection space but not the stability of the estimates reached by the fitting algorithm [7] [22]. However, single tree-based methods such as regression trees and MARS, maybe more vulnerable to multicollinearity. If two predictors are highly collinear, MARS or regression trees have to make an arbitrary knot or split selection that reduces the residual sum of squares the most. This can profoundly affect all subsequent selections and the final predictions [7]. Ensemble methods are less known to be subject to multicollinearity in terms of predictions, because the variability in predictions can be alleviated by implementing a large number of trees. Therefore, the effect of multicollinearity likely varies among the SL methods.

There are numerous popular ways of measuring multicollinearity of a data set such as variance inflation factor, condition number and condition index. We decided to calculate the condition number of the data X because it gives a one-number summary of multicollinearity per data set. A condition number is the ratio of the largest eigenvalue of $\boldsymbol{X'X}$ to the smallest one [1]. We use the `kappa` function in R's `base` package for computations.

5.3 Heteroscedasticity

One of the assumptions of MLR is homoscedasticity, which means the error variance σ_ϵ^2, and hence the variability in Y, is the same across all observations. Equivalently, it indicates that the variance in Y is the same across all y_i's. The violation of this condition is called heteroscedasticity, and can take on numerous structures. We focus on measuring one very common structure, which is that variance increases as the mean increases. In MLR, the presence of heteroscedasticity makes $\hat{\beta}$ less efficient. Worse, data points in the region with high variance have more impact on the unweighted least-squares criterion than data points in the low-variance region [12], which can lead to estimated regression models that predict means with excessively high variance in regions where the variance is low. Ruth and Loughin [26] also discovered the negative impact of heteroscedasticity on tree-based regression methods, and potentially on ensemble methods such as RF and boosting. We anticipate most of the candidate SL methods we investigate in this project tend to produce more variable predictions in the high variance area, but to unknown extents.

To quantify heteroscedasticity, we adopted the measurement proposed by Gelfand called 'Standard Deviation (SD) Ratio' [12]. We first fit the data with a default RF model and computed prediction residuals (out-of-bag errors). Then the ratio is computed as the ratio of the average of the absolute residuals for the largest 10 percent of the predicted responses $\hat{Y}$, to the same average for the smallest 10 percent of $\hat{Y}$. We decided to use RF instead of MLR to capture potential non-linearity in the mean trend, and interactivity among predictors. We use the default RF because the default version usually yields satisfactory performance and makes the measurement easy to compute. Obviously, this ratio is close to 1 for homoscedastic data, and it increases when the variance increases with the mean.

5.4 Signal Strength

The signal strength measures the relative amount of variability in Y attributable to changes in the true mean $f(X)$ compared to the error variance (noise) in Y. It is a crucial data property as it directly reflects the difficulty of identifying the mean trend in Y for any SL method. With a clear signal and low noise, bias plays a relatively more important role than variance, and flexible regression models may be clearly supported when needed. But as the proportion of noise increases (i.e. the signal strength becomes lower), it is harder for SL methods to distinguish between the actual signal and the noise, thus only simpler models are supported.

For a given data set with a linear trend, one can estimate signal strength by calculating the traditional F-statistic, which is the ratio of the mean square regression (which measures the variability explained by the linear model) and the mean squared error (which measures the variability the linear model failed to catch). However, since we have no prior knowledge of the shape of $f(X)$, using linear models might not be an effective way to separate more

complicated signals from the noise. Instead, RF is a suitable candidate for measuring signals because it is known to have flexible shape in arbitrary dimensions, it can automatically adapt to potential unknown interactions, and its default version often yields satisfactory performance. Thus we decided to measure signal strength by calculating the ratio of mean square regression and mean squared error by fitting a default RF model. The ratio, i.e. the F-like statistic, is a unitless measure of the strength of the signal presented by Y in a given data set. In practice, the root mean squared prediction error can be obtained from the out-of-bag error.

5.5 Non-linearity

Non-linearity refers to the scenario where the true mean trend of Y deviates from the linear hyperplane. Some MLR-based SL methods explicitly assume linearity and can be expected to produce biased predictions when the linearity assumption is violated. Beyond that, it is conceivable that some methods might outperform others under more extreme non-linear situations. Knowing the extent of non-linearity of data sets has a practical benefit for users because they can make informed decisions in choosing the right SL methods in terms of model complexity and interpretability.

We measure nonlinearity by comparing RSS from fits of methods that do versus do not assume linearity. The obvious choice for modeling under linearity is MLR. For a similar nonlinear fit, recall that GAM with smoothing splines can capture non-linear mean trends well, yet maintain the assumption of additivity. We therefore measure non-linearity by taking the ratio of the mean square error obtained by fitting the MLR model, to the one from fitting GAM. Naturally, the larger the value, the more obvious the non-linear mean trend is.

5.6 Interactivity

Interactivity refers to the scenario where predictors explain the variability in Y in a non-additive manner. Different SL methods can have dissimilar performance in the presence of interactivity. It is easy to see that interactivity can potentially hurt the predictive performance of additive models, such as MLR and GAM. Projection-based methods, such as PPR and single-layer ANN models, attempt to capture interactivity through non-linear (e.g. activation) functions. We expect tree-based methods to be less vulnerable to interactivity because the recursive partitioning process allows different variables and split locations to influence the fitted function in different regions of X.

To measure interactivity, we focus on second-order interactivity because the principle of hierarchy suggests that second-order tends to be more important than higher orders [13]. We measure interactivity by comparing RSS from fits of the same class of methods, that do versus do not allow second-order interactions. Recall that one of the major features of

MARS is detecting interactions automatically if the maximum level of interaction d in the algorithm is set to be greater than 1. Therefore, MARS is a suitable SL method for detecting interactivity. Specifically, we can fit the data on MARS twice with the specifications of the variables acting additively ($d = 1$), and interactively ($d = 2$). Then the F-like statistic can be obtained by dividing the mean squared error obtained using MARS with $d = 1$, by the one with $d = 2$. The F-like statistic measures how much MSE is reduced due to allowing the second-order interactions. Clearly, the larger the statistic, the more significant the presence of second-order interactivity is.

5.7 Sparsity

We define sparsity as the proportion of important variables among all variables in X. The importance of this property comes from the bias-variance tradeoff. When an SL method is applied to a set of variables that is too large, the predictions may have low bias but potentially high variance. If it is applied to a set that is too small, the predictions may have lower variance but higher bias. Obviously, SL methods with implicit or explicit variable selection schemes can take advantage of the sparse situations and produce competitive predictions, if they can find the right balance between bias and variance.

Let p' be the number of variables out of p that are important in capturing the mean trend in Y. We cannot simply measure sparsity as p'/p. Because p' is unknown and must be estimated. We want to estimate p' in an objective and automated procedure. Ideally it should not only clearly divide variables into 'important' and 'unimportant' groups, but also adapt to non-linearity and interactivity. However, the existing numeric variable importance measures associated with flexible SL methods (e.g., random forest and boosting) provide no clear 'line' with which to separate important variables from the rest. On the other hand, LASSO is known to serve as a variable selection scheme. We can leverage it to measure sparsity by fitting each data set with LASSO (where λ is chosen to minimize the CV error), and calculating the proportion of coefficients that do not equal to zero. LASSO gives a clear decision boundary on the importance of each variable, which makes it easy for us to summarize sparsity into one numerical value per data set. We recognize that this measurement is imperfect as LASSO cannot detect and fit non-linear and interactive data automatically, but we presently cannot find any better and relatively simple alternative.

Chapter 6

Study Methodology

Our proposed methodology is to use measurable data properties to understand the predictive performance of 12 candidate regression SL methods discussed in Chapter 3, on a suite of benchmark data sets. Because of its sequential flow and customizability, we have built an embryonic 'pipeline' in R to streamline the process.

The overview of the pipeline is as follows. The inputs of the pipeline are user-specified sets of data and a user-specified set of SL methods. There are many customizable arguments and parameters (e.g. the random seed, the number of train/test splits, the number of CV folds and the grids for hyperparameter tuning, specific arguments related to each SL method), but we have pre-specified all of them and consider them as the default options. After the input collection, we first generate random 5-fold CV partitions for all input data sets. For each fold, we apply the corresponding SL method to the training data, and the RMSPE is calculated based on the test set. For SL methods that require hyperparameter tuning, we use either the built-in CV or an internal 5-fold CV under the framework of `caret` to select the optimal values of hyperparameters. For each data set we record the RMSPEs for each SL method and the optimal combinations of hyperparameters from each fold for methods that require tuning. Within each data set, we compute relative RMSPEs (RelRMSPEs) by dividing all methods' RMSPEs by the minimum RMSPE for that set. Finally, we build a multivariate regression model that simultaneously predicts each method's RelRMSPE using the 7 data properties. We use the model to examine how important each data property is in affecting the RMSPEs for each SL method. The pipeline can be broken down into three main stages: data preparations, obtain predictive performance of SL methods, and modelling predictive performance based on data properties.

We demonstrate this process by applying 12 candidate SL methods described in Chapter 3 to the suite of 42 real data sets previously studies by Chipman et al. [5] and others. Details are given below.

6.1 Preparing Data

The general purpose of the preparation stage is to ensure these data sets are ready to be used to calculate data properties. We assume that data sets have been pre-processed prior to input, including addressing missing values and influential data points, transforming variables, and creating columns of indicators for categorical variables as necessary. All data properties mentioned in Chapter 5 are then calculated accordingly, using the corresponding measurements. In our example, we further examined the calculated measures by investigating anomalous values and amending the calculated properties, e.g., using transformations, where needed.

6.2 Obtaining predictive performance of SL methods

As the first step of the model-building pipeline, we randomly generate a 5-fold CV to compute RMSPEs for each pre-processed data set. Then within each training set, we create another 5-fold CV for hyperparameter tuning if needed; otherwise, the SL method is applied to the training data and RMSPEs are calculated on each of the 5 folds.

We noticed that many of the variables in our 42 data sets are binary, and some have very few 1's. This creates high likelihood that some CV training sets have variables with constant values, which interferes with fitting many of the SL methods. We implemented a brute-force methodology to address this issue. For each pair of an SL method and a data set, we first created the 5 folds as usual. If any CV training set had a constant value for a variable, then we replaced that training/test split with a separate random 80%-20% draw, independent of the CV folds. In those cases, the splitting scheme became a hybrid of CV and random splitting. This may slightly increase the variability of our RMSPE estimates, but since the same splits were used for all SL methods, we do not expect any bias to enter into the comparisons.

To set up the general framework for model fitting on each training data set, we first created an R function for each SL method that takes the training and test data sets as inputs and produces RMSPEs the outputs. At the heart of each constructed function was an existing R function to fit the SL method to the data. A summary of function names and packages in R is presented in Table 6.1. Note that the last six SL methods (RF, GBM, XGBoost, MARS, PPR and ANN) are tuned and fitted under the framework of the `caret` package using the `train` function. The `train` function works as a wrapper that aggregates a variety of R functions from other packages, so that users can easily select and tune a wide range of SL methods in R[1].

[1]The link to the documentation for all available models (including the ones we applied in this thesis) is http://topepo.github.io/caret/train-models-by-tag.html.

SL method	Package::Function
Multiple linear regression	stats::lm
All subsets regression	leaps::regsubsets
Stepwise regression	stats::step
LASSO	glmnet::cv.glmnet
Relaxed LASSO	relaxo::relaxo
Regression trees	rpart::rpart
Random forest	randomForest::randomForest (via caret)
GBM	gbm::gbm (via caret)
XGBoost	xgboost::xgboost (via caret)
MARS	earth::earth (via caret)
PPR	stat::ppr (via caret)
ANNs	nnet::nnet (via caret)

Table 6.1: A summary table of SL methods with the corresponding function names and packages in R.

We had to make adjustments to some of the SL algorithms in order to fully implement them to all data sets. First, the `regsubsets` function for all-subsets regression cannot handle data sets with more than 49 explanatory variables. Where necessary, we modified the algorithm, by first performing backward stepwise regression using the `step` function to reduce the number of variables to 49. We also set the maximum size of subsets to examine to be 10 to limit the computation needed. Secondly, we tried to implement GAM by using the `gam` function in the `mgcv` package, with the `s` function to build smoothing splines. Although GAM is a powerful tool for modelling non-linear data, we realized that finding the 'optimal' basis dimension for each variable in X via minimizing GCV may not be effective when some variables have skewed distributions (especially when clear clusters are present). We found that the numbers of basis dimensions needed to be set beforehand for a given data set, which likely requires human intervention. If it was set too large, the sum of basis dimensions could be larger than n, which would result in the failure of GAM. An automatic solution could be studied in the future, but we decided to exclude GAM from the list of candidate SL methods for now.

We next set up the framework for hyperparameter tuning. For parametric SL methods including LASSO and Relaxed LASSO, the tuning parameters λ and ϕ were chosen by existing functions `cv.glmnet` and `cvrelaxo` with built-in CVs. For methods that don't have built-in tuning functions in R, such as RF or GBM, we used the grid search algorithm to perform hyperparameter tuning. Specifically, after identifying hyperparameters for an SL method to tune on, the grid search algorithm creates a 'grid' with all combinations of hyperparameters, and finds the one that minimizes some estimate of the generalization error (e.g. the CV error). We thus need to pre-specify all tuning parameter values for corresponding SL methods. A summary of all 'grid' values is presented in Appendix A. Note that for RF, the required parameter `mtry` represents the number of randomly sampled

explanatory variables considered at each split. In this experiment, since the total number of explanatory variables for each data set varies drastically, we provided values for `mtry` indirectly by specifying percentages of variables to be considered at each split instead.

We used the `train` function in the `caret` package to perform 5-fold CV and select optimal hyperparameter values. Inputs of the `train` function are the training data and all combinations of the hyperparameter values we want to select from. The output is the model with optimal hyperparameter values that minimize the CV error. The `predict.train` function in the `caret` package produces predictions by using the best model and it does not require users to specify additional arguments (e.g. in the `predict.gbm` function, users have to specify the argument `n.trees`, but not in the `predict.train` function [21]), and RMSPEs can be easily calculated. We found that ANNs had very poor performance in many data sets and discovered that the default number of iterations allowed by the `nnet` function, 100, was too low. We raised it to 1000 and achieved substantially better results.

After estimating all methods' predictive performance on each data set, we examined the optimal hyperparameter values obtained from each train/validation split for all SL methods that required hyperparameter tuning through the `caret` package, to ensure that the optimal values are not usually on the boundary and no further grid expansion is needed. We adjusted the grid as necessary to reduce the frequency with which boundary values were selected.

6.3 Modelling predictive performance based on data properties

The general flow of the final stage is as follows. We first averaged all RMSPEs obtained from separate CV folds and calculated all RelRMSPEs across all SL methods per data set. Thus we obtained a single measure of each SL method's relative performance on each data set. We then built a multivariate linear regression model regressing these quantities on measured data properties, to analyze how important each property is in affecting the RelRMSPEs for each SL method. The response variables $Y_1, \ldots, Y_{12}$ contained RelRMSPEs for all 12 SL methods, and $X_1, \ldots, X_7$ contained measured data properties for each of the 42 benchmark data sets.

We used using the multivariate linear regression model for several reasons. First of all, we wanted to take the correlations between RMSPEs from different SL methods into account in our standard errors and tests; thus, we did not use 12 univariate regression models. Secondly, since there are 7 explanatory variables (measured properties) but only 42 observations (benchmark data sets) for predicting 12 response variables (SL methods), performing this analysis using more complex SL methods, such as ensemble methods, was not feasible. We also wanted to use a technique that would allow us to cleanly interpret the effects of each variable on the methods' RelRMSPEs.

Formally, let $Y_{ik} \in \mathbb{R}$ be the RMSPE for the k-th SL method on the i-th data set, where $i \in \{1, \ldots, 42\}$, and $k \in \{1, \ldots, 12\}$. Let $x_{ij} \in \mathbb{R}$ be the j-th data property for the i-th data set, where $j \in \{1, \ldots, 7\}$. Let $b_{0k} \in \mathbb{R}$ be the regression intercept for k-th SL method and $b_{jk} \in \mathbb{R}$ be the j-th data property's regression slope for k-th Sl method. Finally, let $(e_{i1}, \ldots, e_{im}) \overset{\text{iid}}{\sim} \mathrm{N}(\mathbf{0}_m, \boldsymbol{\Sigma})$ be a multivariate Gaussian error vector with unknown variance matrix Σ. Then the multivariate linear regression model for RMSPEs over all SL methods has the form [18] of

$$Y_{ik} = b_{0k} + \sum_{j=1}^{p} b_{jk} x_{ij} + e_{ik}$$

where $i \in \{1, \ldots, 42\}, j \in \{1, \ldots, 7\}, k \in \{1, \ldots, 12\}$. The estimation of the coefficient matrix $\boldsymbol{b_{42\times 12}}$ is equivalent to equation-by-equation least squares for the individual responses in (3.4). We can then determine which methods are sensitive to which data properties by examining the estimated coefficients.

We can also conduct the Type II Multivariate Analysis of Variance (MANOVA) test for each variable, by comparing models with and without the variable, while retaining all others in the model. Since all statistical inferences for the multivariate regression model account for the correlation among the response variables, the foundation of the decomposing the sum of squares total (SST) is slightly different than it is for univariate regression. Under the multivariate regression setting, the total variability is measured by the total sum-of-squares-and-cross-products (SSP_T), and it can be decomposed into regression SSP (SSP_R) and residual SSP (SSP_E). Let SSP_H represents the incremental SSP matrix for a hypothesis test, that is, the difference between SSP_R for the model without the tested variable and SSP_R for the model with it. A multivariate test for the hypothesis is then based on the 12 eigenvalues $\lambda_j, j = 1, \ldots, 12$ of $SSP_H SSP_E^{-1}$. We implement here the default choice of test statistic for the `Anova` function, which is called Pillai-Bartlett Trace with the form of $\sum_{j=1}^{m} \frac{\lambda_j}{1-\lambda_j}$. The `car` package uses F approximations to the null distributions of the test statistic [9]. More details are covered in [9] and [25].

Chapter 7

Experimental Study Results

We conducted the experiment applying 12 candidate SL methods on the 42 real-world benchmark data sets. In this chapter, we present results from the experiment, and describe some necessary adjustments to the planned analyses.

7.1 Results of the measured data properties

Summaries of the initial data property measurements are shown in Appendix Table B.1 and Figure B.1. After examining these measurements, we identified some measures with outlying values for some individual data sets. Specifically, the data set Fishery had an extreme value for multicollinearity. We identified that it was because the last column was a perfect linear combination of others, and we addressed it by removing the last column. The Strike data set also had an extreme value for multicollinearity because the 5^{th} column was highly collinear with the 1^{st} column; so we decided to remove the 5^{th} column. The Hatco data set also had an extremely high multicollinearity value because the 8^{th} and 10^{th} columns summed to 1. Hence we removed the 8^{th} column. The Labor data set had a high heteroscedasticity value. We identified that the response variable was presented as a combination regression and classification problem, with around half of the responses at 0 and the other half occupying a range quite distant from zero. We addressed this problem by removing all the observations with $y_i = 0$.

After addressing these anomalies, several of the measures had extreme values and/or highly skew distributions. To reduce the potential for high influence from tail values, we applied log transformations to all measured properties except sparsity. In addition, since the distributions of interactivity and data richness were so highly skewed, we had to take second log transformations on them. Thus the measure of data richness eventually became $log(log(\text{data richness}))$ and the measure of interactivity became $log(log(\text{interactivity}+1))$, where we added 1 to all $log(\text{interactivity})$ values because there exist some small negative values. The final measured data properties are in Appendix Table B.2. The histograms of measured data properties after the adjustments are displayed in Figure 7.1. We can see that

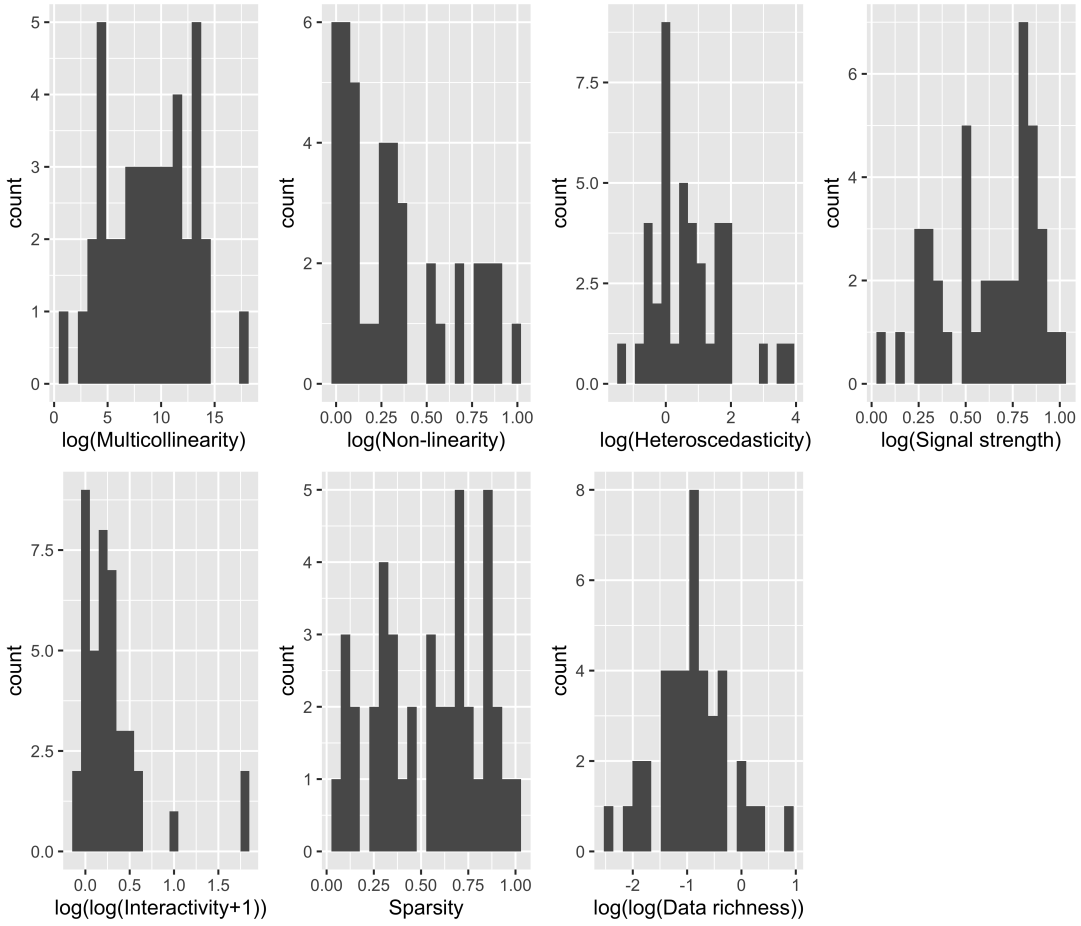

Figure 7.1: Histograms of measured data properties on benchmark data sets after adjustments.

except for two or three clear outliers for interactivity, there are no extreme values in any of the remaining measured data properties.

7.2 Results of predictive performance

After we applied 12 SL methods on the suite of 42 benchmark data sets, we visualized the results by making boxplots of RelRMSPEs for each SL method and each data set. The initial boxplots are displayed in Appendix C.

We first compared RelRMSPEs for all 12 SL methods. From Appendix Figure C.1, we observed that the 5 MLR-based methods gave nearly identical relative performance. This observation is further supported by the correlation matrix of the RelRMSPEs for all 12 SL methods shown in Figure 7.2. We can see that the correlations of RelRMSPEs between MLR-based methods are rounded to 1. This is an indication that the flexibility of an SL method is a hugely important characteristic in affecting its relative performance, and that

adjustments to strictly linear models have comparatively very limited impact. Hence we can simplify comparisons of SL methods by dropping all but one linear regression method from future comparisons. We decided to use LASSO as the representative because of its popularity and it yielded the lowest median RelRMSPE within the group.

We also drop regression trees from further analysis. While the ideas behind regression trees are revolutionary, it is well-known that individual trees can have unstable performance. This is seen clearly from Appendix Figure C.1. Therefore, there were 7 SL methods remaining in our full analyses.

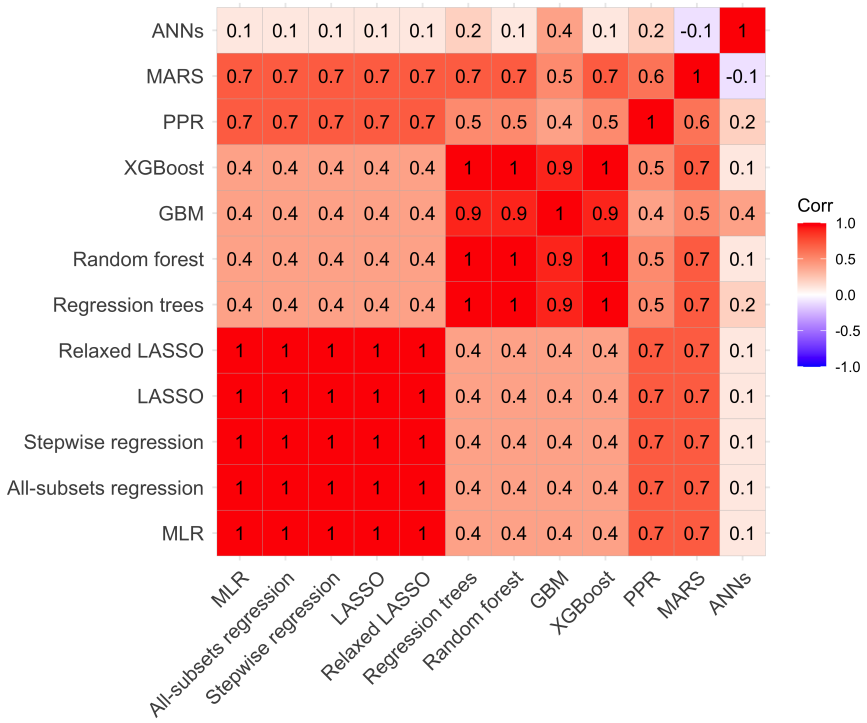

Figure 7.2: Correlation plot of RelRMSPEs for all 12 SL methods

The boxplot of RelRMSPEs from the remaining 7 SL methods is shown in Appendix Figure C.2. Recall that when RelRMSPE equals one, the corresponding SL method outperforms all other candidate SL methods for that data set. We can clearly see that LASSO performed badly in many data sets, and it seldom yielded the best performance among all 7 SL methods. ANNs and PPR had competitive performance and they had the lowest 'worst case' RelRMSPEs among all SL methods. All remaining methods have much better overall performance but the differences are obscured due to the scale. Figure 7.3 shows a portion of that plot, where we have cut off the display at RelRMSPE= 2 so that we can more clearly see how methods performed on the majority of the data sets. Here we see that ensemble methods, especially boosting methods like GBM and XGBoost, tended to be among the

best for a large proportion of the data sets as they often won in the competitions. They had the lowest median and 75-percentile values as well. Both ANNs and MARS had fairly competitive relative performance, while PPR had generally slightly larger RelRMSPEs. LASSO (and all MLR-based methods) clearly performed the worst.

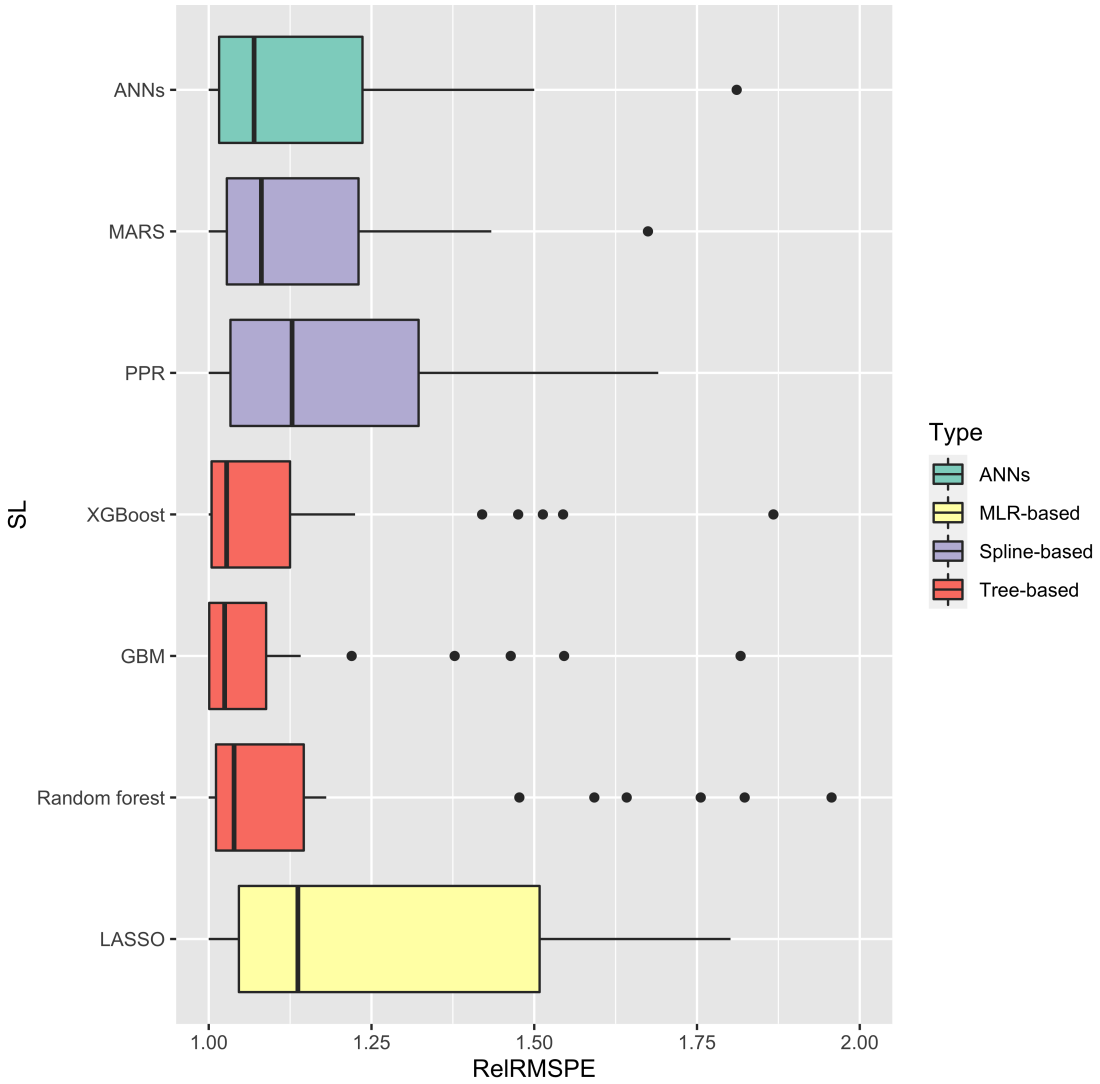

Figure 7.3: Boxplot of RelRMSPEs from 1 to 2 for 7 SL methods. Number of RelRMSPEs larger than 2 for each method (that are not plotted) were the following: LASSO 7, Random forest 3, GBM 2, XGBoost 1, PPR 2, MARS 3, ANNs 2.

From these figures, it is obvious that magnitudes of the worst RelRMSPEs must vary substantially among data sets. For instance, the relative performance of random forest is competitive for most of the data sets, but it has an extreme value in one particular data set. To see more clearly how methods reacted differently to different data sets, we show the boxplot of RelRMSPEs per benchmark data set in Appendix Figure C.3. We observed that

the Budget data set [2] had an extremely large outlier, that could be potentially influential for the multivariate regression modelling stage. Further investigation of this data set revealed some very strange relationships among variables that did not seem natural. Consulting the original source for these data [2] led to our conclusion that something was corrupted with the version that we accessed, so we removed it from further consideration.

The boxplot of RelRMSPEs for all remaining 41 data sets is shown in Figure 7.4. It is interesting to see that some data sets such as Alcohol, Medicare and Wage are predicted roughly equally well regardless of the choice of SL method. On the other hand, data sets such as Tecator, Cpu, and Insur, have extremely variable RelRMSPEs across candidate SL methods compared to other data sets. The distribution of RelRMSPE for the Tecator data set is interesting. The relative performance for all methods other than ANNs is extremely poor. This phenomenon is a strong indication that a data set can be highly sensitive to the choice of the SL method. Tecator is so highly sensitive to the selection, it provided the worst case performance for 4 out of 6 eligible SL methods, and the second worst case for the remaining two. We investigated an intermediate source for this data [27], but could not find any reason to omit the data set, so we allowed it to remain. However, we anticipate that those RelRMSPE values will be highly influential in estimating the coefficients of the multivariate linear regression model.

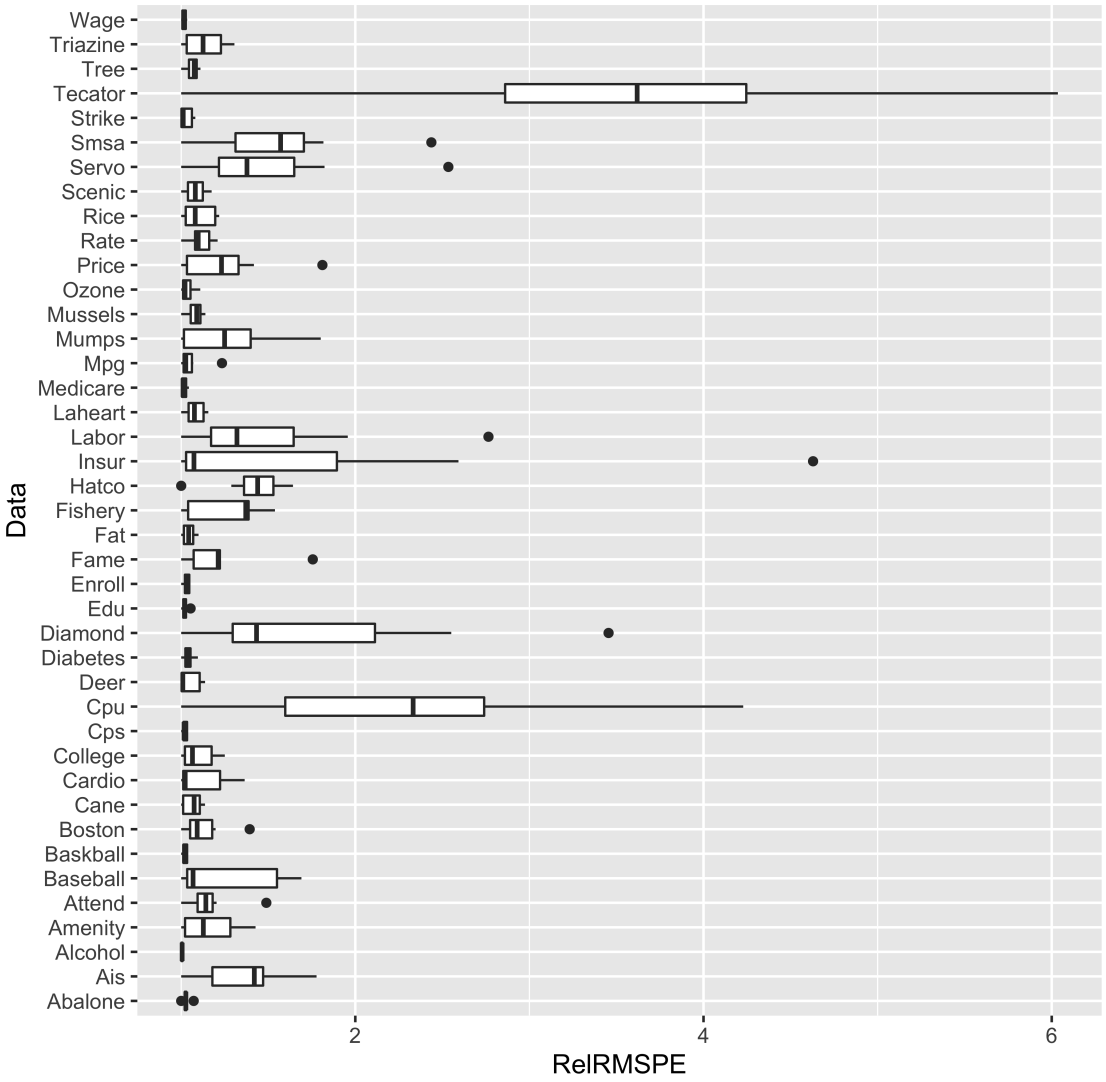

Figure 7.4: Boxplots of the RelRMSPEs for 41 benchmark data sets.

7.3 Multivariate regression model results

Next, we study whether each data property affects SL methods' relative performance and whether the effects are the same across all SL methods. To answer these questions, we first produced scatterplots of the RelRMSPEs versus each data property for each SL method. The full plots are in Appendix D, and we highlight here one property with a clear and consistent pattern and one with no clear trends. Figure 7.5 depicts the plots of RelRMSPE against non-linearity values for each SL method. Recall that the larger the measured value, the more obvious the non-linear mean trend is in $f(X)$. Generally speaking, with one clear exception in LASSO, SL methods' performance is relatively stable on data sets with low non-linearity. In data sets with high non-linearity, LASSO often performed badly, but most

other SL methods (e.g. random forest, GBM and ANNs) also seem to struggle compared to other methods on certain data sets.

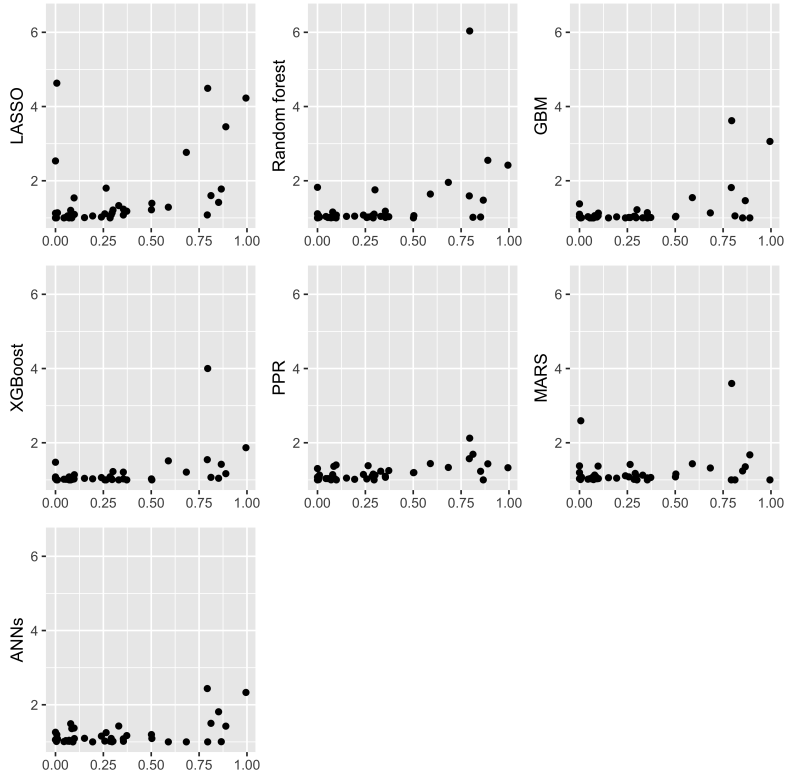

Figure 7.5: Scatterplots of the RelRMSPEs for each SL method versus non-linearity.

Figure 7.6 shows the relationships between measured multicollinearity and RelRMSPEs. Recall that we measure multicollinearity using the logged condition number, and the higher the measured value, the more severe the multicollinearity problem in a data set. From the scatterplots, we can immediately see that there is one data set with the lowest measured multicollinearity value but the highest RelMSPEs for many SL methods (the Tecator set mentioned previously). Without that data point, we cannot see any significant mean change in RelRMSPE in general. However, we anticipate that the coefficient estimates for multicollinearity in the multivariate linear regression model will be heavily influenced by this data set and indicate a negative linear association between multicollinearity and RelRMSPEs. The true trends in these plots are often hard to discern due to the high influence and the small number of data sets, and the estimated coefficients in the multivariate linear regression model may not be trustworthy.

Next, we built the multivariate regression model using the 7 measured data properties to model RelRMSPEs of 7 candidate SL methods. The estimated coefficients with standard errors are shown in Table 7.1, where all statistically significant coefficient estimates are shaded in grey. As expected, most of the estimated coefficients for multicollinearity were

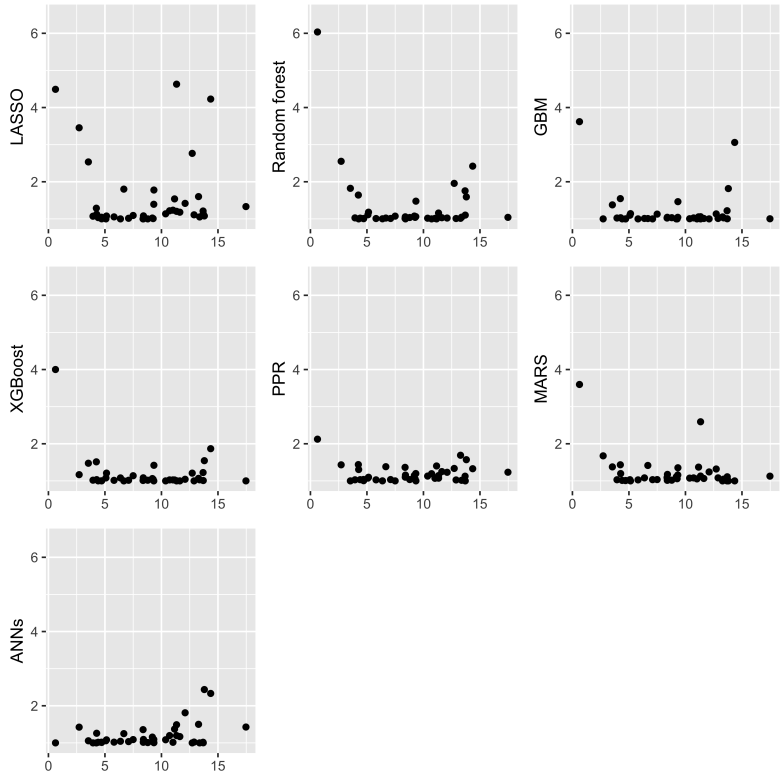

Figure 7.6: Scatterplots of the RelRMSPEs for each SL method versus multicollinearity.

negative and statistically significant due to the outlier. Coefficients estimates for both non-linearity and interactivity showed that these properties could significantly affect the relative performance of ensemble methods. Nearly all estimated coefficients for signal strength are positive, which might be counter-intuitive. More details are discussed in Chapter 8. Finally, the coefficient estimates of sparsity and data richness were hardly significant.

In addition, we used the `Anova` function in the `car` package to perform a Type II MANOVA hypothesis test for testing the significance of main effects (i.e. measured data properties). All test statistics and p-values are displayed in the last row of Table 7.1, and the statistically significant ones are shaded in grey. We can see that 3 out of 7 data properties are statistically significant in modelling RelRMSPEs for all 7 SL methods. These data properties, non-linearity, heteroscedasticity and interactivity, are the ones that have been widely studied. We can see the effect of signal strength from the Appendix Figure D.4, but surprisingly it is not a statistically significant covariate.

	(Intercept)	Multicollinearity	Non-linearity	Heteroscedasticity	Signal strength	Interactivity	Sparsity	Data richness
LASSO	0.62 (0.51)	-0.099 (0.037)	0.10 (0.55)	0.126 (0.119)	1.782 (0.770)	1.047 (0.365)	0.249 (0.603)	-0.18 (0.22)
Random forest	1.39 (0.46)	-0.126 (0.033)	1.00 (0.49)	0.082 (0.107)	0.822 (0.689)	0.474 (0.326)	-0.027 (0.540)	-0.06 (0.19)
GBM	1.09 (0.28)	-0.052 (0.021)	0.70 (0.30)	0.117 (0.066)	0.094 (0.424)	0.407 (0.201)	0.091 (0.332)	-0.05 (0.12)
XGBoost	1.24 (0.28)	-0.070 (0.020)	0.56 (0.30)	0.087 (0.065)	0.375 (0.421)	0.195 (0.200)	0.003 (0.330)	-0.04 (0.12)
PPR	1.20 (0.13)	-0.024 (0.009)	0.34 (0.14)	0.004 (0.030)	0.320 (0.194)	-0.019 (0.092)	-0.247 (0.152)	-0.05 (0.05)
MARS	1.11 (0.29)	-0.071 (0.021)	-0.13 (0.31)	0.034 (0.067)	1.156 (0.433)	0.106 (0.205)	-0.093 (0.339)	-0.07 (0.12)
ANNs	0.92 (0.17)	0.013 (0.013)	0.43 (0.18)	0.117 (0.040)	-0.064 (0.257)	0.006 (0.122)	-0.121 (0.202)	-0.05 (0.07)
Overall test	NA	2.1 (0.080)	3.0 (0.018)	3.6 (0.007)	1.2 (0.356)	2.5 (0.043)	0.6 (0.744)	0.2 (0.969)

Table 7.1: A summary table of estimated coefficients from the multivariate regression model with their standard errors. Entries in main body are coefficient (standard error). The last row contains the results of the Type II MANOVA Pillai tests in the form 'test statistic (p-value).' The coefficient estimates and test statistics that are statistically significant at $\alpha = 5\%$ level are shaded in grey.

Chapter 8

Discussion

In this chapter, we further investigate and interpret the results shown in Chapter 7. Then we reflect on the methodology we adapted in this project and discuss some current limitations and areas for improvement.

8.1 Discussion of results

In Section 7.2, we found some interesting trends in RelRMSPEs for each SL method by examining Figure 7.3. MLR-based SL methods all performed similarly and they struggled with many data sets. This phenomenon was expected because their bias can be heavily dependent on the properties of data sets. They can have low bias when the relationship between Y and X is linear and additive, but very high bias when modelling more complicated relationships, especially when data density and signal strength allow more flexible methods to be estimated with low variability. There are seven data sets where LASSO had RelRMSPEs more than double the best result. After cross-referencing with the measured data properties, these data sets all have either high non-linearity or high interactivity values, which is coherent with our hypothesis.

Recall that the MSPEs can be decomposed into squared bias and variance of the predictions, and that RelRMSPEs are formed in competition relative to the RMSPEs of other methods on each data set. The nature of the estimated coefficients in Table 7.1 and Appendix Figures D can be traced to how each SL method's bias and variance change as the data properties change. If a measure increases as data sets become more 'difficult' to predict, then methods that adapt most quickly to higher levels of the property would have negative slopes. They might be increasingly dominant among all SL methods as the measure increases. Other less effective adapters would have positive slopes as they are surpassed by more other methods and by greater amounts.

We can use this rationale to better understand the plots and coefficients, particularly for those properties where stronger patterns were observed. Recall that some data properties, such as multicollinearity and heteroscedasticity, may interfere with an algorithm's ability

to find a suitable solution, i.e. to select the optimal member of its class. These two data properties mainly affect the variance of predictions, but we speculate that multicollinearity could relate to a lesser degree to bias, if it interferes systematically with identifying the proper variables. For multicollinearity, if we ignore the outlier on the left-most edge of most plots in Appendix Figure 7.6, results matched with our expectation that methods with no explicit or implicit variable selection scheme in their algorithms (e.g. spline-based methods and ANNs) might be more resistant to increasing levels of multicollinearity. ANNs showed a slightly increasing trend on the right, which might be an indication that the algorithm has difficulty settling on weight values and might give unstable results. Ensemble methods reduced both bias and variance by combining a large-enough number of trees (or iterations). LASSO is known to be vulnerable to multicollinearity, and it was reflected in these 41 data sets.

On the other hand, non-linearity could mainly affect the bias of a model if it fails to capture the pattern well. As shown in Appendix Figure D.2, LASSO's RelRMSPEs increased drastically as non-linearity got larger, because more flexible methods are better able to adapt it. When measured non-linearity is low, all methods performed similarly, with some occasional exceptions (2 in LASSO, 1 in MARS). Also, there were positive associations between SL methods' RelRMSPEs and non-linearity, which may suggest that non-linearity can manifest in different forms, and different methods are better with some forms than others.

Signal strength could affect predictive performance of SL methods by affecting the variance of predictions. As mentioned, when the signal strength is weak, variance dominates bias for all SL methods because it is harder for more flexible models to isolate the true mean trend from random noise. As the signal gets stronger, flexible methods can more easily adapt to complicated shapes, thus variance plays a relatively smaller role in affecting MSPE compared to bias. RelRMSPEs, in this case, can be considered as the reflections for each SL method's capability to capture the mean trends in Y correctly. However, since different shapes could be better predicted by different methods, we anticipate that the variance of RelRMSPEs from the competition would increase for all methods. From Appendix Figure D.4, it is obvious that all methods performed similarly when the signal was poor; when signal got stronger, no one method dominated (and the variance of RelRMSPEs increased), which is consistent with our expectation.

8.2 Discussion of limitations

Our study did not go completely as planned for a variety of reasons, some of which we anticipated and some of which were unexpected. First, we knew that our measurements of data properties might not all be perfect. Our goal at this stage was to develop 'quick and dirty' measures that are sensitive and specific to the features we measured. But we have not

tested sensitivity and specificity to the respective properties, and we are not convinced that these measured values reflect the true levels of these properties. For instance, we settled on measuring sparsity by using LASSO, even though we were aware of LASSO's severe limitation to linear, additive models. It would be nice to use an explicit feature selection technique based on flexible regression SL methods, but we are not immediately aware of well-established techniques for this. We also settled on measuring interactivity by only considering the two-factor interactions, while in reality deeper interactions can possibly also affect an SL method's predictive performance. We hope to study and improve the generality of these measurements in the future. Potentially, we could build multiple MARS models with several maximum degrees of interactions, and compare the best-fitting version to the degree-1 MARS model. We will rigorously validate the current measurements, and add more possible data properties in future studies. Further contributions to the pool of data properties from researchers will be strongly encouraged.

Even though we obtained all predictive performances of SL methods through two 16-core virtual machines, the process of tuning boosting methods (e.g. GBM and XGBoost) was still very time-consuming and challenging. As a result, the process of hyperparameter tuning was limited to the 5-fold CV on parsimonious grids. As we see in the histograms in Appendix E, there remain optimal tuning parameter values that lie on the boundaries for GBM and XGBoost. Since a fair comparison requires that each SL method be represented by its best possible version, identifying appropriate ranges for tuning parameters is another area for future work.

The biggest limitation we have in project is that we cannot draw any firm conclusions based on the Table 7.1 for two reasons. First of all, due to the existence of influential data points in both X (i.e., measured data properties) and Y (i.e., RelRMSPEs for 7 SL methods) in the plots in Appendix D, the multivariate linear regression model using the least-squares criterion may not be the best approach for capturing and reflecting the true relationship between X and Y. For example, we have seen from the Figure 7.6 that due to the data point on the left (the Tecator data set) with the extreme RelRMPSE value, the estimated multicollinearity coefficients for nearly all SL methods were negative and statistically significant, whereas the rest of the data sets did not suggest any meaningful impact of multicollinearity other than for ANNs. Since most SL methods experienced their worst RelRMSPEs from Tecator, its extreme influence directly affected the confidence we have in the results of the multivariate linear regression model. There are similar problems with outliers in data properties as well. As shown in both Figure 7.1 and Appendix Figure D.5, the two extreme interactivity values on the right flattened the estimated coefficients, while the rest of the data sets showed a positive linear association between interactivity and RelRMSPEs.

Another reason for low confidence in the final regression analysis is we used only 41 data sets and we have identified only 7 data properties. Even though some properties appeared

to show a pattern of relationship with RelRMSPEs, the trends were still hard to discern due to the lack of data. The small sample size also limits our capability to use more complex SL methods to predict RelRMSPEs. For example, data richness is a property that may interact with other properties, like non-linearity and interactivity, by allowing more complex SL methods to perform relatively better. But it might not be as important as a main effect in a regression analysis. We would need more data sets to enable modeling interaction effects or nonlinear trends. Also, these are not the only properties that could affect the predictive performance of SL methods. The effects of unmeasured properties remain in the errors from the multivariate regression analysis and hinder our ability to understand all relationships.

To drive more insights out of the results, we tried to perform two post-hoc analyses. We first examined the relationships between the median RelRMSPEs across all methods for each data set, as seen in Figure 7.4, and the respective measured data properties. The median RelRMSPE, in this case, represents how sensitive a data set is to the choice of SL method, where higher values mean that performance of different methods was quite different on those data. Based on Figure 8.1, we can see that even though the effects of influential data points were not completely removed, the medians are sensitive to changes in most data properties, which validated our idea that data properties could potentially affect an SL method's predictive performance. We just cannot adequately understand the relationships with the present data. The second analysis we conducted was to perform log transformations on all RelRMSPEs to reduce the influence of extreme values. However, the extreme values remained and the results were substantively similar to what we obtained without the transformation. In particular, nearly all RelRMSPEs for the Tecator data set were still influential. Further study of the source of that data is needed. However, since it is unrealistic to eliminate all data sets with unusual characteristics, we speculate that a better alternative could be using robust regression methods to limit influence of individual data sets.

In summary, we are not convinced that we have measured properties in best ways, and further research needs to be conducted to better study and validate the structures of these properties. Also, more data sets are needed to both surface the trends between RelRMSPEs and data properties, and make more complicated SL methods for predictions affordable.

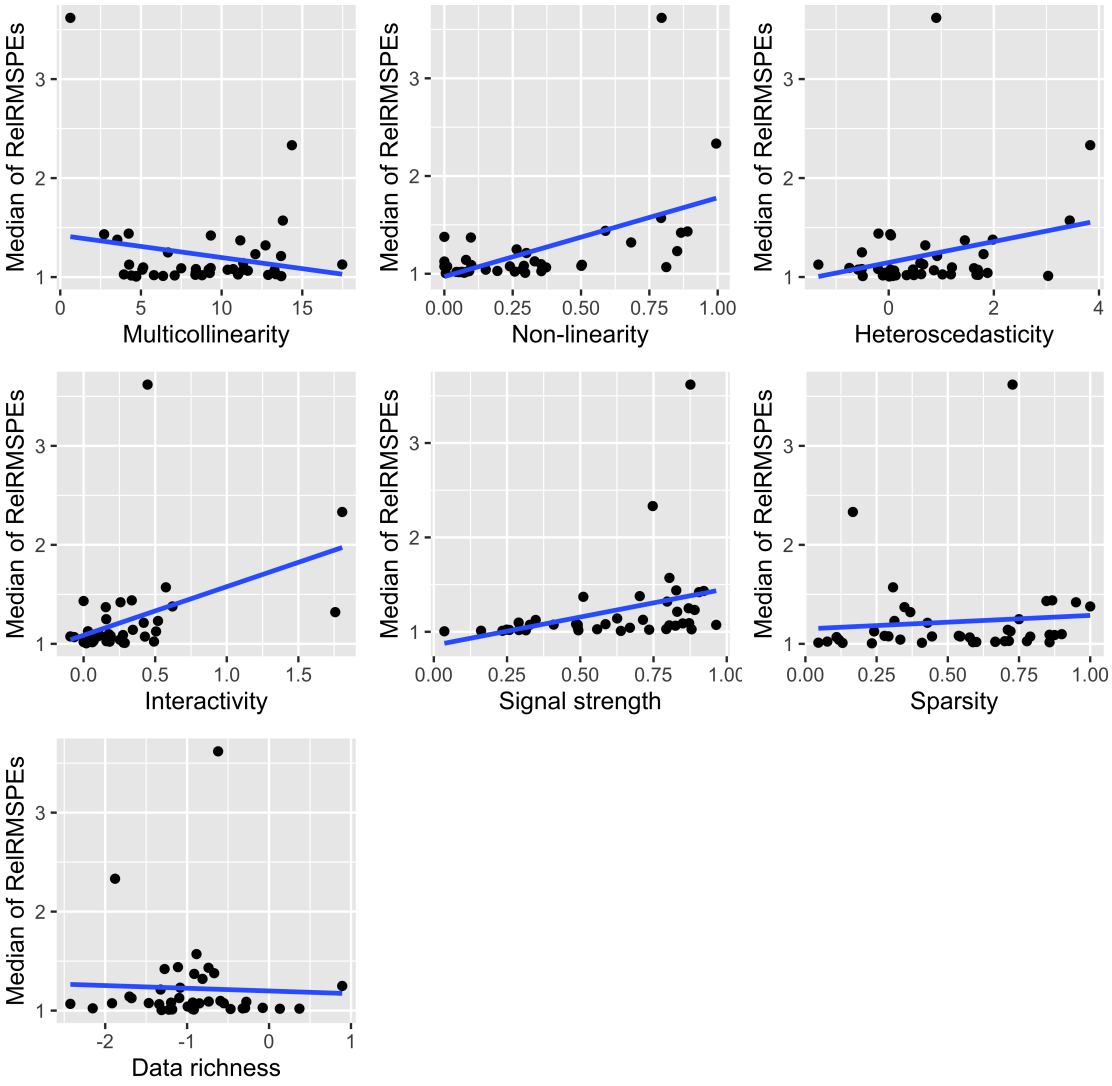

Figure 8.1: Scatterplots of the medians of RelRMSPEs across all 7 SL methods versus each data property. The blue straight lines represent the linear regression models.

Chapter 9

Conclusion and Future Works

In this thesis, we explored the potential to estimate the prediction performance of SL regression methods using data characteristics. We demonstrated the process on 42 benchmark data sets [5], and measured 7 properties on each data set that potentially affecting an SL method's predictive performance. Through cross-validation, we measured the RelRMSPEs of 12 well-known SL methods. Finally, we combined the relative performance measures and the data properties into a multivariate regression model to investigate the importance of each data property in predicting relative predictive performance for all SL methods. We further analyzed the rationale behind the experimental results and discussed the limitations with the current measurements and methodologies.

We have successfully established the initial stage of the pipeline with the automatic functionalities of measuring data properties, producing predictive results, and performing in-depth analyses. The pipeline is easy to use, and easy to extend if a user wants to compare their newly developed SL method or data set with what we currently have in the 'arena.' The pipeline is also customizable as users could effortlessly change the settings or arguments. From the analysis perspective, we have successfully unlocked the potential of using data properties to predict the relative performance of an SL method, such that users could better understand the right selections of the SL methods for their data sets.

By comparing SL methods' predictive performance and building a multivariate regression model, we found some interesting insights. There is no SL method that always produces the best predictions across all data sets, although we informally suggest that ensemble methods are preferred when the computation power is permitted. We found that the medians of RelRMSPEs across all SL methods were highly sensitive to changes in measured data property values, which supported our initial belief that these properties related to prediction performance.

However, we are inhibited to produce reliable estimated coefficients and predictions of the RelRMSPEs, due to the lack of sample size (we only have 41 benchmark data sets) and the presence of extreme values in both RelRMSPEs and measured data properties. Further

studies are required to better validate the current methodology and introduce more data properties and data sets (and potentially SL methods) into the current system.

For future works, we recommend reinforcing the flexibility and usability of our proposed pipeline. Specifically, we only considered the $n > p$ scenario, whereas $n < p$ or even $n << p$ cases are becoming more frequent, especially in -omics fields, and they worth studying further. Furthermore, it would be useful to develop an R Shiny Application in the future to improve the user experience. We project that the new Web App interface would have the options to select and filter existing results, upload a new pre-processed data set or user-defined SL function for comparisons, and display results in both tabular and graphical formats. Additionally, we propose to include more data sets into the current pipeline in two ways: import data sets from well-known data repositories such as UCI [8] and StatLib [20], and simulate data sets that represent interesting properties. We would like to have more extensive coverage for each of the data properties, such that RelRMSPEs could be well estimated for a newly introduced data set.

Bibliography

[1] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons, 2005.

[2] C Andrea Bollino, Federico Perali, and Nicola Rossi. Linear household technologies. *Journal of Applied Econometrics*, 15(3):275–287, 2000.

[3] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[4] Tianqi Chen and Carlos Guestrin. 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.

[5] Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[6] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015.

[7] Richard D De Veaux and Lyle H Ungar. Multicollinearity: A tale of two nonparametric regressions. In *Selecting models from data*, pages 393–402. Springer, 1994.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[9] John Fox, Michael Friendly, and Sanford Weisberg. Hypothesis tests for multivariate linear models using the car package. *The R Journal*, 5(1):39–52, 2013.

[10] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[11] Jerome H Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.

[12] Sharla Jaclyn Gelfand. Understanding the impact of heteroscedasticity on the predictive ability of modern regression methods. 2015.

[13] Michael Hamada and Jeff Wu. *Experiments: planning, analysis, and parameter design optimization*. Wiley New York, 2000.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[15] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[16] Trevor J Hastie and Robert J Tibshirani. *Generalized additive models*, volume 43. CRC press, 1990.

[17] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[18] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*, volume 5. Prentice hall Upper Saddle River, NJ, 2002.

[19] Hyunjoong Kim, Wei-Yin Loh, Yu-Shan Shih, and Probal Chaudhuri. Visualizable and interpretable regression models with good prediction power. *IIE Transactions*, 39(6):565–579, 2007.

[20] Charles Kooperberg. Statlib: an archive for statistical software, datasets, and information. *The American Statistician*, 51(1):98, 1997.

[21] Max Kuhn. The caret package, Mar 2019.

[22] Isabella Morlini. On multicollinearity and concurvity in some nonlinear multivariate models. *Statistical Methods and Applications*, 15(1):3–26, 2006.

[23] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1):36, 2017.

[24] Fikri Öztürk and Fikri Akdeniz. Ill-conditioning and multicollinearity. *Linear Algebra and Its Applications*, 321(1-3):295–305, 2000.

[25] Calyampudi Radhakrishna Rao, Calyampudi Radhakrishna Rao, Mathematischer Statistiker, Calyampudi Radhakrishna Rao, and Calyampudi Radhakrishna Rao. *Linear statistical inference and its applications*, volume 2. Wiley New York, 1973.

[26] Will Ruth and Thomas Loughin. The effect of heteroscedasticity on regression trees. *arXiv preprint arXiv:1606.05273*, 2016.

[27] Hans Henrik Thodberg. Tecator meat sample dataset. statlib datasets archive, 2015.

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

Appendix A

The table of pre-specified tuning parameter values for SL methods that require hyperparameter tuning using the `caret` package

SL method	Grid values
Random forest	`mtry` = seq(0.1,0.9,0.1)
GBM	`interaction.depth` = c(1:4,6,8), `n.trees` = c(100,500,1000,2000,4000,8000), `shrinkage` = c(0.001,0.005,0.01,0.05,0.1), `n.minobsinnode` = c(3,7,10)
XGBoost	`nrounds` = c(100,500,1000,2000,4000), `max_depth` = seq(2,8,2), `eta` = c(0.001,0.005,0.01,0.05,0.1), `gamma` = 0, `colsample_bytree` = c(0.7,0.8,0.9), `min_child_weight` = 1, `subsample` = 0.8
MARS	`degree` = 1:3, `nprune` = c(2,3,seq(5,25,5))
PPR	`nterms` = 1:4
ANNs	`size` = seq(1,20,2), `decay` = seq(0.0001,0.9,length.out = 9)

Table A.1: The table of pre-specified tuning parameter values for SL methods that require hyperparameter tuning using the `caret` package.

Appendix B

Results from data property measurements on 42 benchmark data sets

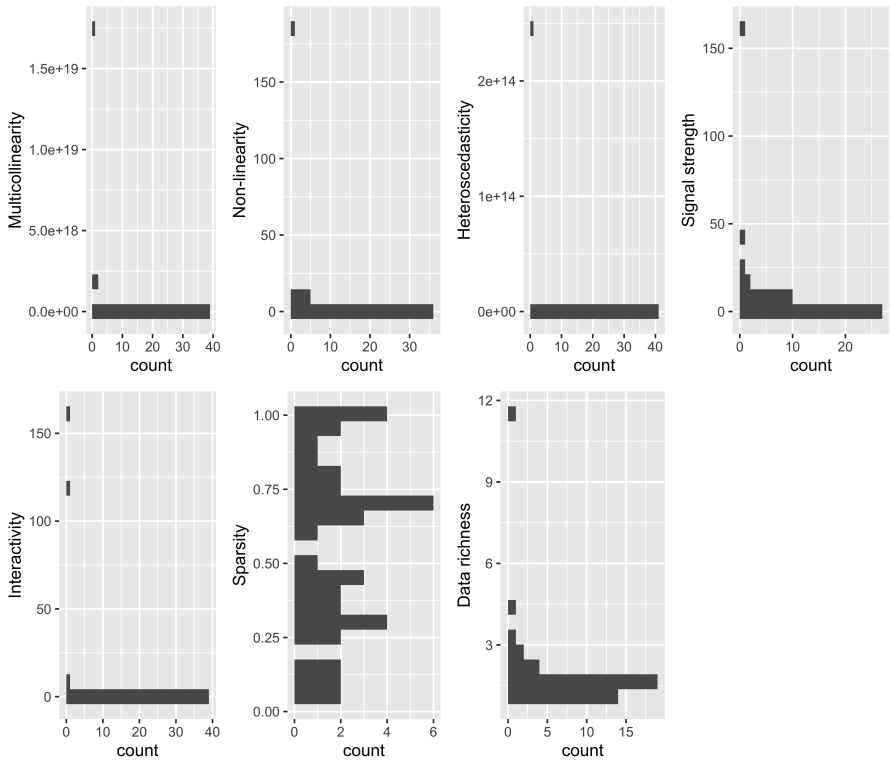

Figure B.1: Histograms of measured data properties on benchmark data sets before any adjustments.

Name	n	p	Multicollinearity	Non-linearity	Heteroscedasticity	Signal strength	Interactivity	Sparsity	Data richness
Abalone	4177	8	51.5150582	1.1001176	3.26485498	1.26089339	1.02810912	0.7	2.5253308
Ais	202	12	11296.8768	7.45437125	1.04442868	15.6234233	1.34169094	0.95	1.32231305
Alcohol	2467	18	110.897927	1.00517014	1.01062425	0.10867681	1.01921314	0.26666667	1.30900398
Amenity	3044	21	38835707.1	1.49322803	1.92835843	2.2240373	1.03121303	0.8	1.39682991
Attend	838	9	83736.8	1.08612352	1.82490327	1.77756019	1.50750581	0.71052632	1.19951752
Baseball	263	20	585937.644	5.31759035	2.36467416	4.81510026	0.93873663	0.109375	1.09247623
Baskball	96	4	6550.42678	1.04635298	1.40707471	0.31829626	1	0.6	3.13016916
Boston	506	13	11177.6348	2.01334193	0.47302262	6.01144856	1.12162979	0.71428571	1.6144016
Budget	1729	10	14298.0646	3.16249459	6.03136191	161.2833	4.98606793	1	2.10755783
Cane	3775	9	31876.8212	1.010332	1.03540632	0.67726881	1.1845999	0.52631579	1.15843993
Cardio	375	9	4328.74985	1.09311456	5.3043618	0.36995897	1.88766898	0.05769231	1.12323648
College	694	24	111878.404	1.59305674	1.11078261	4.26946406	1.23317961	0.69230769	1.29913529
Cps	534	10	1200.83602	1.05745294	1.14268668	0.51920331	1.06509971	0.64705882	1.4807145
Cpu	209	7	1715836.98	185.861411	46.2039776	6.10573669	162.057128	0.13888889	1.16490335
Deer	654	13	889659.646	1.4197657	0.61013781	2.53590083	1.39550465	0.13043478	1.34270562
Diabetes	375	15	10558.1574	1.17860533	6.55544548	0.96930717	1.33418926	0.11764706	1.44836144
Diamond	308	4	15.1571478	9.05579644	1.0300727	19.8649084	1	0.92307692	1.61205267
Edu	1400	5	331.677796	1.06778853	1.61401006	0.22619854	1.02865381	0.66666667	4.2581956
Enroll	258	6	645265.008	1.24075815	1.85553131	3.5688371	1.07505948	0.71428571	2.52311251
Fame	1318	22	875561.67	1.42862888	2.51249938	6.25028309	1.68520725	0.46428571	1.30482546
Fat	252	14	4412.93988	1.39888108	0.93544442	1.82993626	1.03561579	0.33333333	1.48432366
Fishery	6806	14	1.7462E+19	1.10751216	4.35981292	1.5765065	1.18233806	0.39130435	1.49355892
Hatco	100	13	1.6214E+18	2.43536886	0.8435345	3.86365499	1.49192884	0.93333333	1.38949549
Insur	2182	6	83983.2475	1.00691036	5.44932098	42.8661821	1.70649816	0.78947368	1.53282272
Labor	2953	18	253801.969	2.87724569	2.4536E+14	22.9232985	121.296857	0.36842105	1.55880724
Laheart	200	16	9856.04503	1.31549187	1.58802147	0.5721881	1.10993812	0.29166667	1.25905523
Medicare	4406	21	109.087963	1.07907266	1.08156354	0.31702193	1.02183167	0.40909091	1.49116578
Mpg	392	7	60975.2751	1.55001166	2.78099662	6.53721269	1.18876296	0.77777778	2.10940882
Mumps	1523	3	789.654744	1.35943736	0.60120296	4.57787647	1.18669255	0.75	11.5053535
Mussels	201	4	1800.12709	1.10930936	5.0654216	5.1578374	1.37225966	1	2.13318248
Ozone	330	8	394238.07	1.34681118	5.50652857	2.77110705	1.22071099	0.66666667	2.06449694
Price	159	15	178562.417	6.74828746	6.07264528	7.30554805	1.97649764	0.3125	1.40203808
Rate	144	9	170.510254	1.54866444	3.33047522	0.46117269	1.213089	1	1.73707294
Rice	171	15	45223.1721	2.0044632	0.59957116	1.0838518	1.07887004	0.27777778	1.3531711
Scenic	113	10	4479.33884	1.41043956	0.81767911	0.88544731	1.23077138	0.46153846	1.4828249
Servo	167	4	34.2789032	1	7.23727942	1.85357603	2.36408296	1	1.66829039
Smsa	141	10	978876.443	4.8261553	31.2955869	3.74520437	2.17396847	0.30769231	1.51043344
Strike	625	5	2.226E+18	1.07660867	20.3733177	0.34647458	1.37902835	0.04545455	1.35874245
Tecator	215	10	1.86830206	4.87398816	2.47046543	4.63743141	1.75626971	0.72727273	1.71097572
Tree	100	8	161.728861	1	0.55942443	0.95366733	0.91542623	0.44444444	1.77827941
Triazine	186	28	71.3549703	1	0.26242628	0.66410417	1.93287418	0.24137931	1.20518588
Wage	3380	13	81.6814019	1.07714182	0.89523723	1.02244913	1.00915045	0.85714286	1.86833664

Table B.1: A summary table of 42 benchmark data sets' measured properties before any adjustments.

Name	n	p	log(Multicollinearity)	log(Non-linearity)	log(Heteroscedasticity)	log(Signal strength)	log(log(Interactivity+1))	Sparsity	log(log(Data richness))
Abalone	4177	8	3.94187416	0.09100627	1.18321534	0.55749804	0.02734403	0.7	-0.0764793
Ais	202	12	9.33228158	0.86585052	0.04347002	0.90524069	0.25768465	0.95	-1.2751734
Alcohol	2467	18	4.7086102	0.00514355	0.01056821	0.03598636	0.01885208	0.23333333	-1.3120536
Amenity	3044	21	17.4748507	0.33030992	0.65666909	0.71397675	0.03027292	0.72	-1.0959997
Attend	838	9	11.3354338	0.0792944	0.60152698	0.62584192	0.34391341	0.71052632	-1.7041915
Baseball	263	20	13.2809687	0.8119449	0.86064024	0.80383628	-0.0653072	0.109375	-2.425353
Baskball	96	4	8.78728548	0.04429957	0.34151288	0.29038167	0	0.6	0.13198136
Boston	506	13	9.32167017	0.50331338	-0.7486121	0.87168518	0.10865958	0.85714286	-0.7361291
Budget	1729	10	9.56787947	0.68379393	1.79697284	0.9940559	0.95806495	0.90909091	-0.2936601
Cane	3775	9	10.3696344	0.01022634	0.03479393	0.32787671	0.15649514	0.54385965	-1.916818
Cardio	375	9	8.37303406	0.0851828	1.66852946	0.24884938	0.4918524	0.07692308	-2.1523199
College	694	24	11.6251679	0.37227597	0.10506482	0.82505433	0.19028633	0.57692308	-1.3405608
Cps	534	10	7.09077327	0.05433144	0.13338223	0.31417806	0.06115946	0.58823529	-0.9351557
Cpu	209	7	14.3554116	0.99461965	3.83306589	0.74730452	1.80631123	0.16666667	-1.8796854
Deer	654	13	13.6985943	0.29565843	-0.4940704	0.63846437	0.28762415	0.13043478	-1.2218425
Diabetes	375	15	9.26465405	0.15153955	1.88029608	0.49172638	0.253342	0.11764706	-0.993083
Diamond	308	4	2.71847222	0.88957349	0.02962938	0.92153243	0	0.84615385	-0.7391737
Edu	1400	5	5.80416401	0.06348498	0.4787218	0.25786905	0.02785927	0.66666667	0.37076703
Enroll	258	6	13.3774164	0.19404116	0.61817108	0.79344554	0.06987674	0.71428571	-0.0774284
Fame	1318	22	13.6826209	0.30021521	0.92127803	0.83078363	0.41995203	0.42857143	-1.3239985
Fat	252	14	8.39229639	0.28514295	-0.0667336	0.66940381	0.03439777	0.33333333	-0.9289728
Fishery	6806	14	11.1597023	0.09707538	1.44793456	0.51044279	0.15485947	0.34782609	-0.9133904
Hatco	100	13	4.2395944	0.58938458	-0.1943353	0.8275626	0.3365221	0.86666667	-1.1118777
Insur	2182	6	11.3383726	0.00686294	1.69549101	0.9640211	0.42816772	0.78947368	-0.8507115
Labor	2953	18	12.7212837	0.68335548	0.69578392	0.79662279	1.75755458	0.36842105	-0.8121088
Laheart	200	16	9.19584026	0.23982806	0.46248889	0.40906021	0.09921551	0.29166667	-1.4681049
Medicare	4406	21	4.69215456	0.07327835	0.07840771	0.23415611	0.02136687	0.40909091	-0.9173958
Mpg	392	7	11.0182237	0.35484356	1.02280936	0.87994862	0.1594906	0.77777778	-0.2924833
Mumps	1523	3	6.67159582	0.26440154	-0.5088227	0.87003909	0.15800331	0.75	0.89315002
Mussels	201	4	7.49561255	0.09853821	1.62243737	0.84970596	0.27494538	0.875	-0.27758
Ozone	330	8	12.8847102	0.25750542	1.7059344	0.73511465	0.18184934	0.66666667	-0.3217401
Price	159	15	12.0926935	0.85181426	1.80379431	0.89039082	0.51958301	0.3125	-1.0849255
Rate	144	9	5.13879544	0.35428232	1.203115	0.28955077	0.17661363	0.9	-0.5938423
Rice	171	15	10.7193649	0.50111332	-0.5115406	0.58615369	0.07317075	0.27777778	-1.1958367
Scenic	113	10	8.40723074	0.29100117	-0.2012853	0.48571088	0.18866896	0.53846154	-0.9315339
Servo	167	4	3.5345301	0	1.97924537	0.70298019	0.62078625	1	-0.6698226
Smsa	141	10	13.7941607	0.79279573	3.44347709	0.80430352	0.57467569	0.30769231	-0.8857696
Strike	625	5	6.37196282	0.0711574	3.03279736	0.16127123	0.27867601	0.04545455	-1.1823431
Tecator	215	10	0.62503003	0.79482921	0.90440657	0.87591066	0.44672993	0.72727273	-0.6216384
Tree	100	8	5.08592124	3.26E-13	-0.5808468	0.49130563	-0.0925161	0.44444444	-0.5522619
Triazine	186	28	4.267667	-4.44E-16	-1.3377851	0.34775979	0.5062199	0.24137931	-1.6786068
Wage	3380	13	4.40282634	0.07161714	-0.1106665	0.49351113	0.0090676	0.85714286	-0.469926

Table B.2: A summary table of 42 benchmark data sets' measured properties after adjustments.

Appendix C

Predictive performance from SL methods

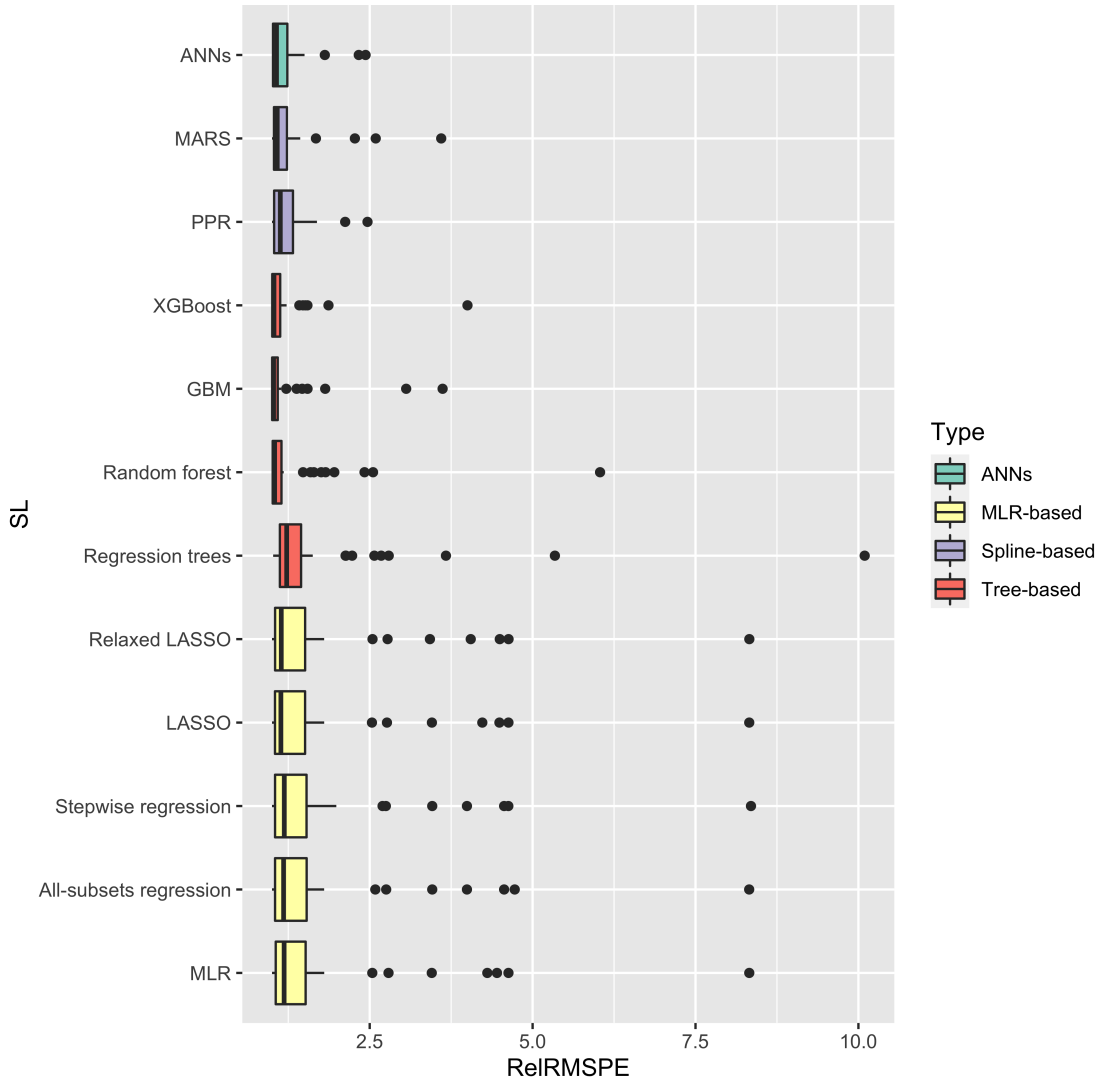

Figure C.1: The boxplot of the RelRMSPEs for each of the 12 SL methods across the full 42 benchmark data sets.

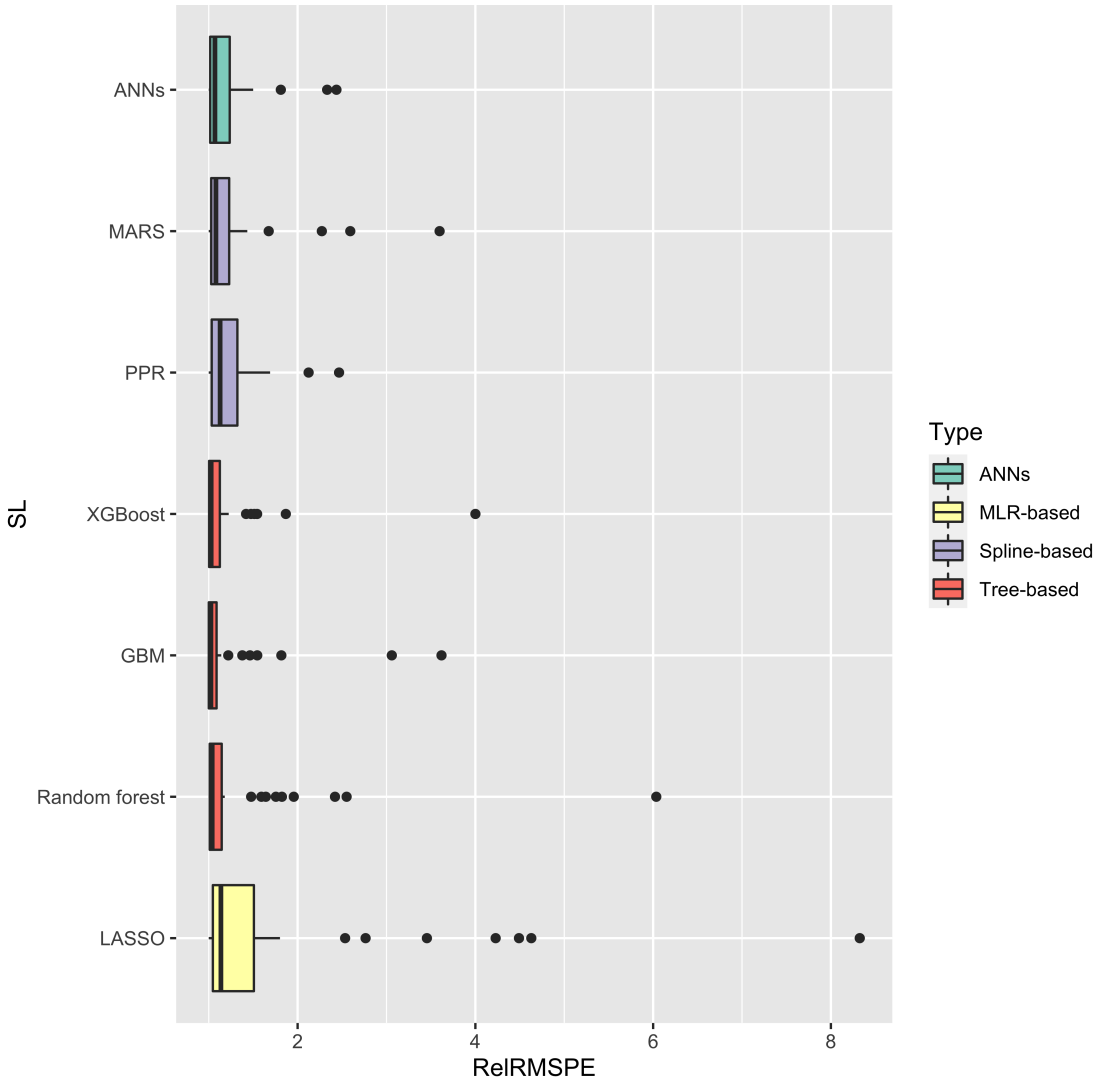

Figure C.2: The boxplot of the RelRMSPEs for 7 SL methods across all 42 benchmark data sets (no cutoff).

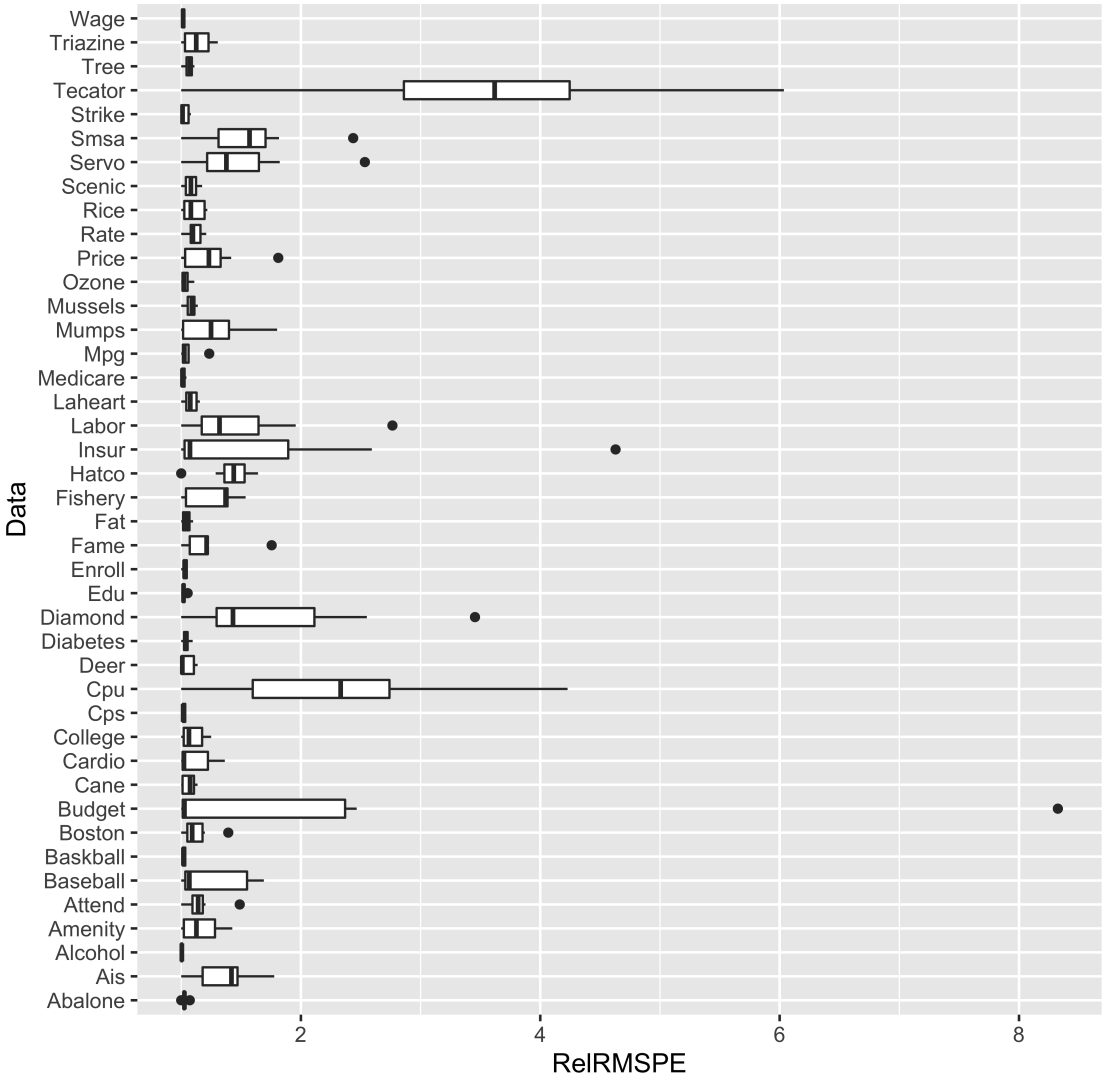

Figure C.3: The boxplot of the RelRMSPEs for all 42 benchmark data sets across 7 SL methods.

Appendix D

Scatterplots of 7 measured data properties versus RelRMSPEs for the final 7 SL methods

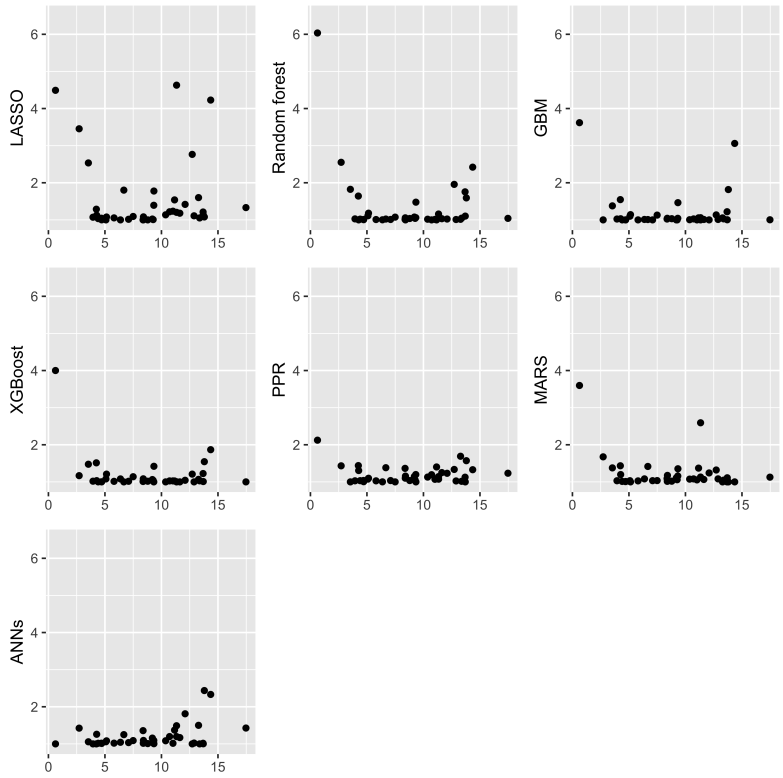

Figure D.1: Scatterplots of the RelRMSPEs for each SL method versus log(multicollinearity).

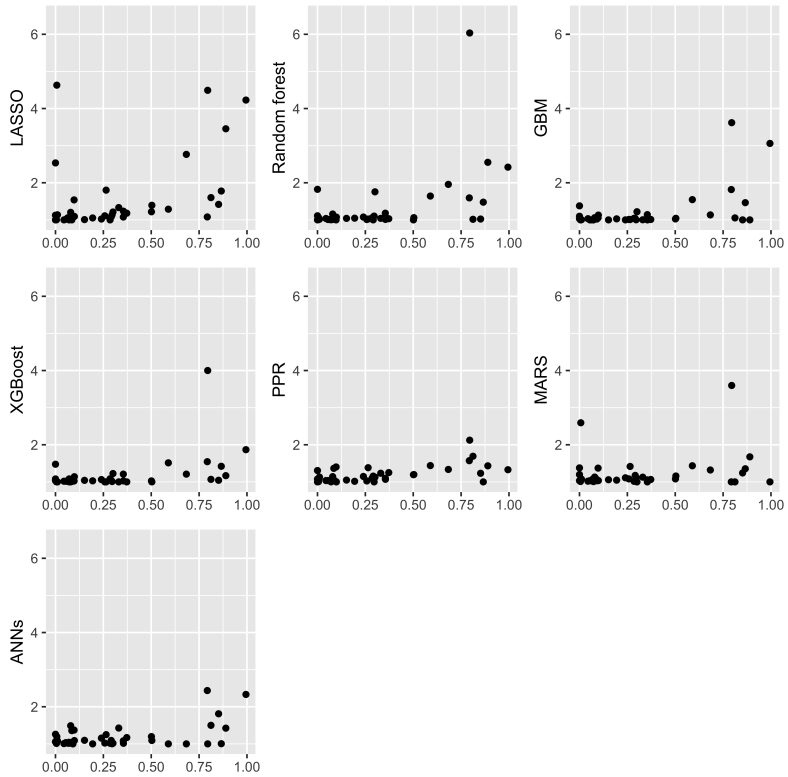

Figure D.2: Scatterplots of the RelRMSPEs for each SL method versus log(non-linearity).

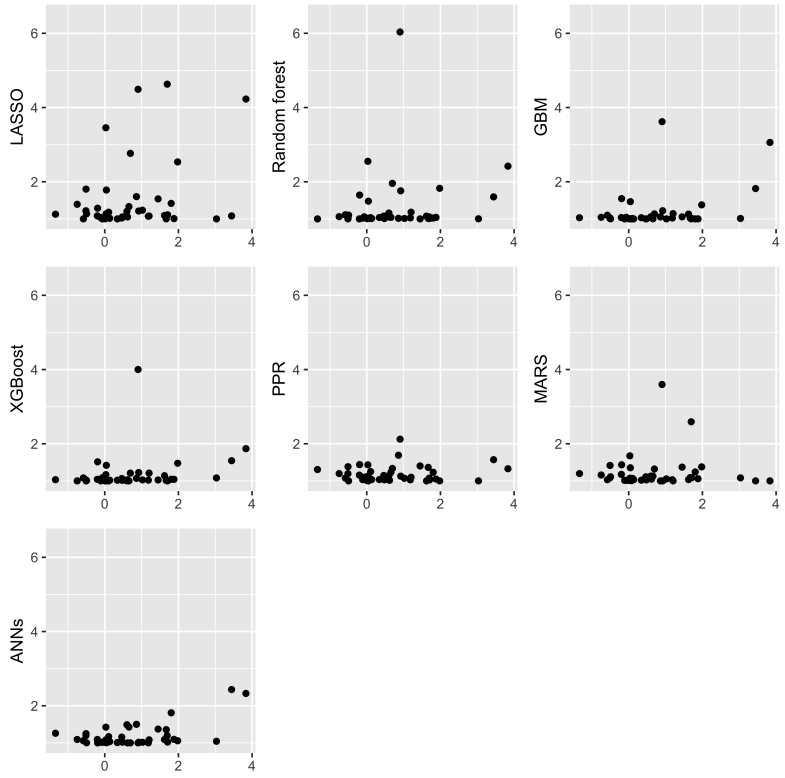

Figure D.3: Scatterplots of the RelRMSPEs for each SL method versus log(heteroscedasticity).

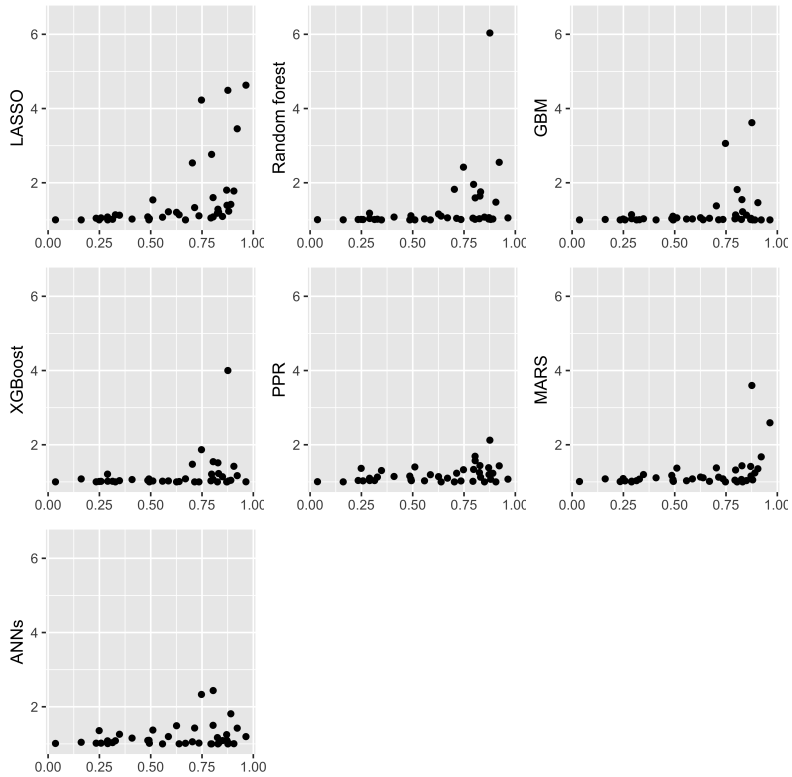

Figure D.4: Scatterplots of the RelRMSPEs for each SL method versus log(signal strength).

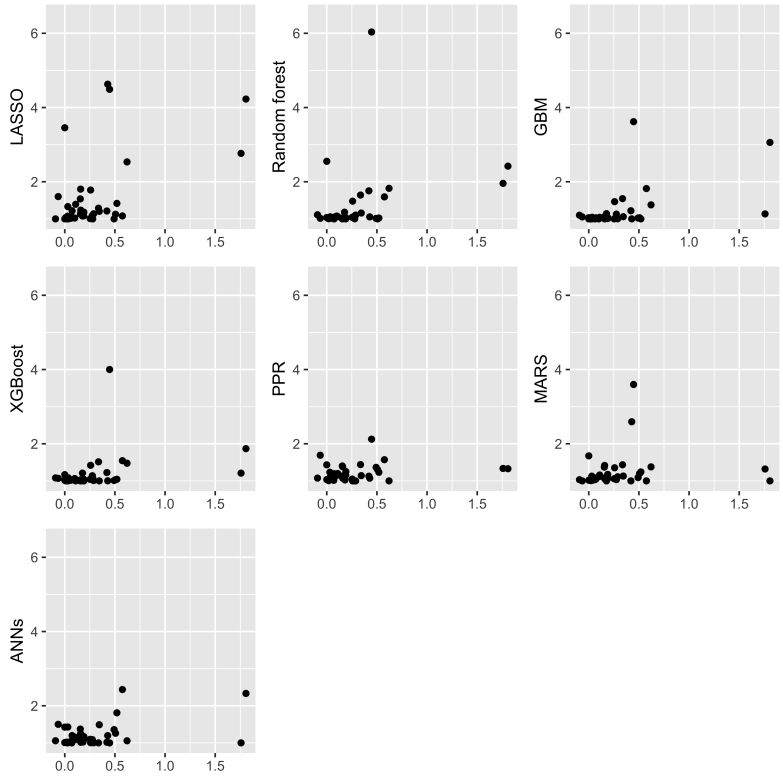

Figure D.5: Scatterplots of the RelRMSPEs for each SL method versus log(log(interactivity+1)).

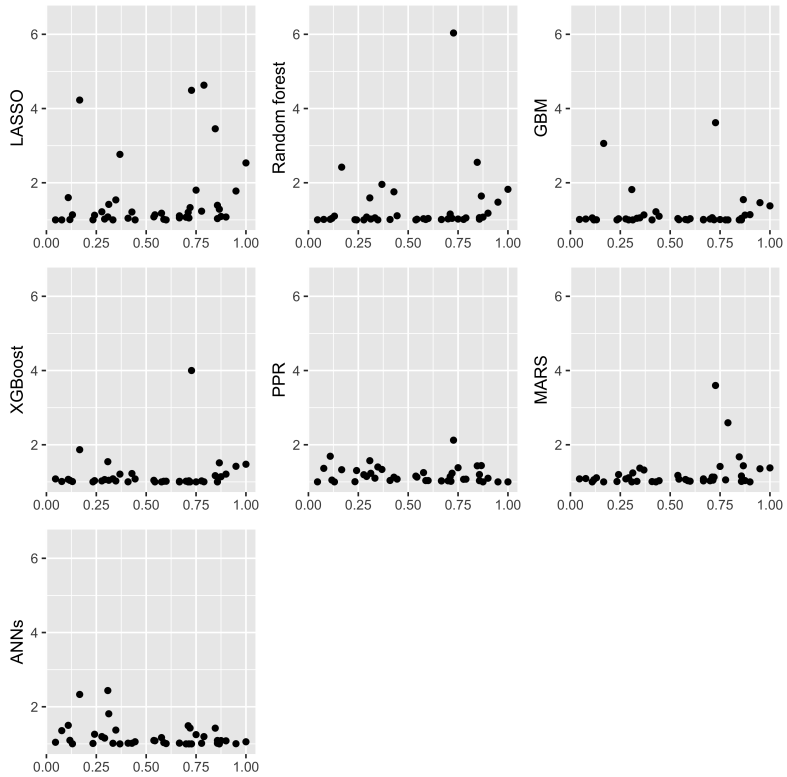

Figure D.6: Scatterplots of the RelRMSPEs for each SL method versus sparsity.

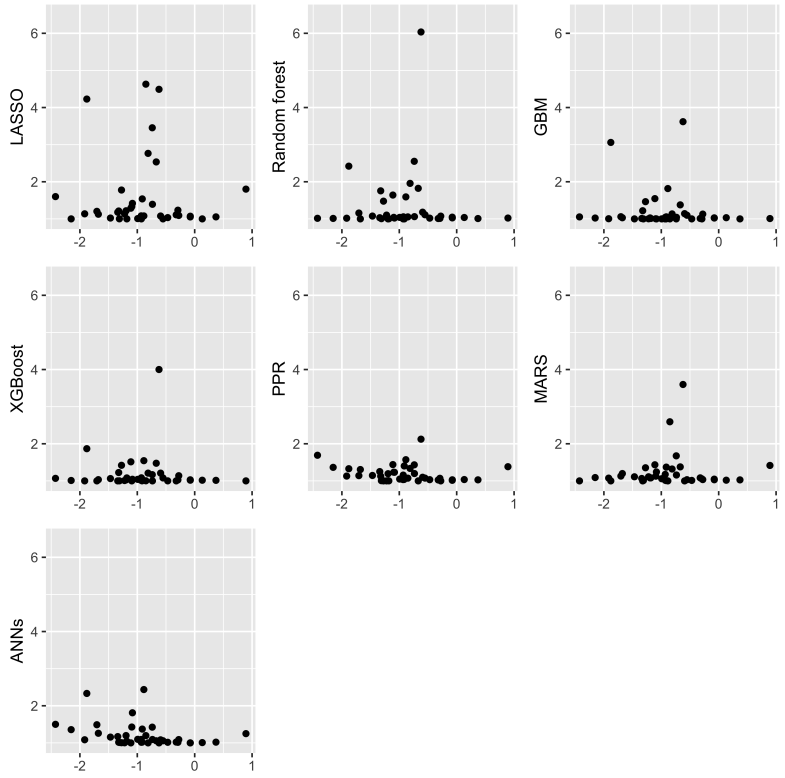

Figure D.7: Scatterplots of the RelRMSPEs for each SL method versus log(log(data richness)).

Appendix E

Histograms of optimal hyperparameter values for 6 SL methods tuned using the `caret` package

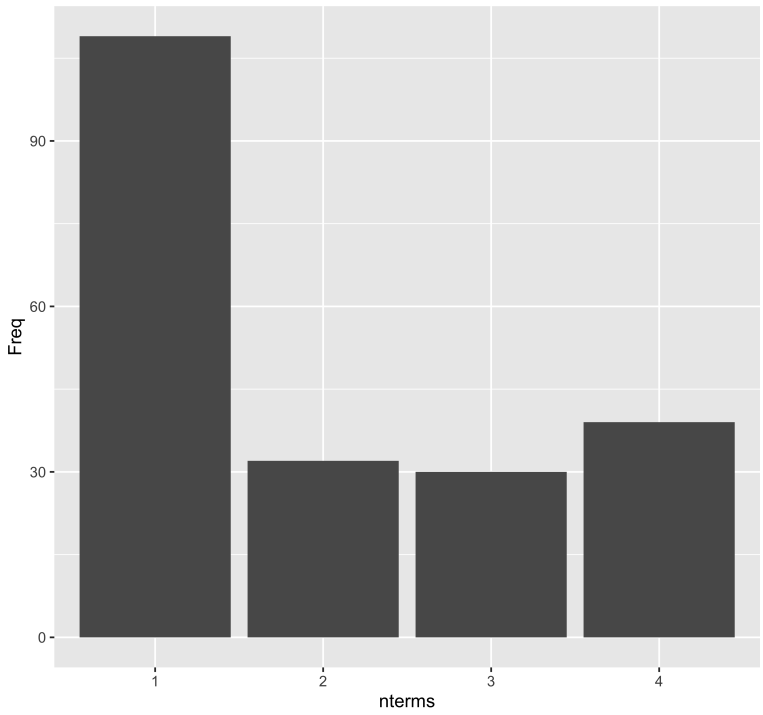

Figure E.1: Histogram of optimal hyperparameter values for PPR.

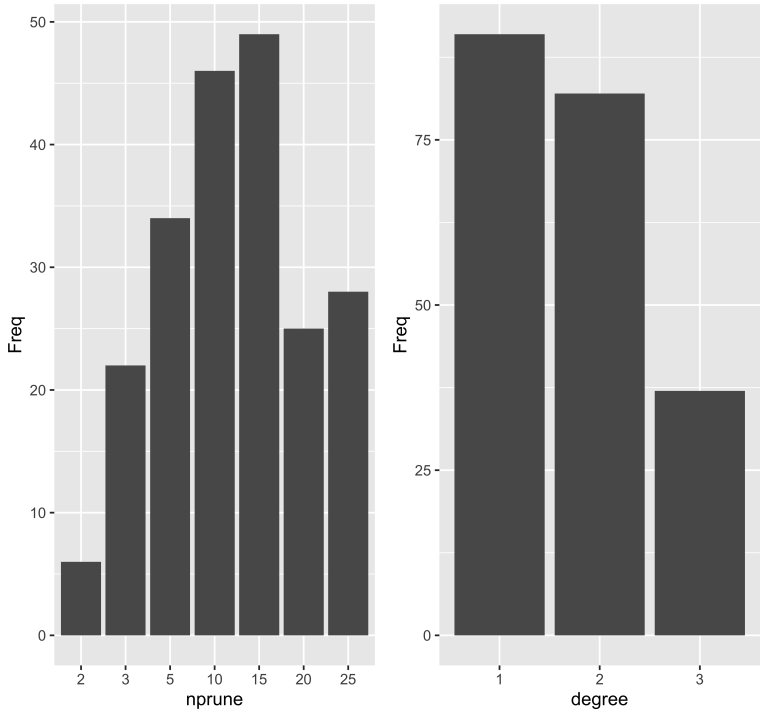

Figure E.2: Histogram of optimal hyperparameter values for MARS.

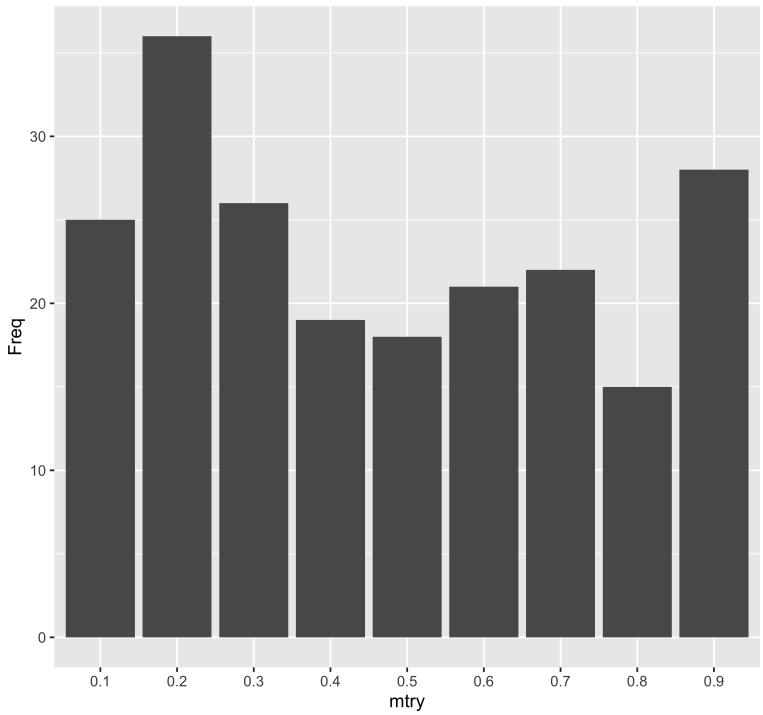

Figure E.3: Histogram of optimal hyperparameter values for RF.

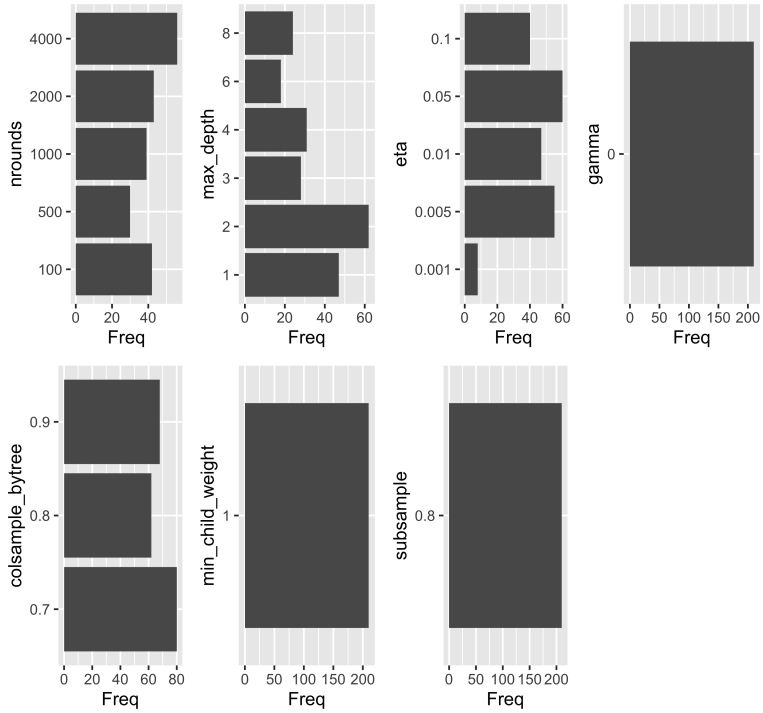

Figure E.4: Histogram of optimal hyperparameter values for XGboost.

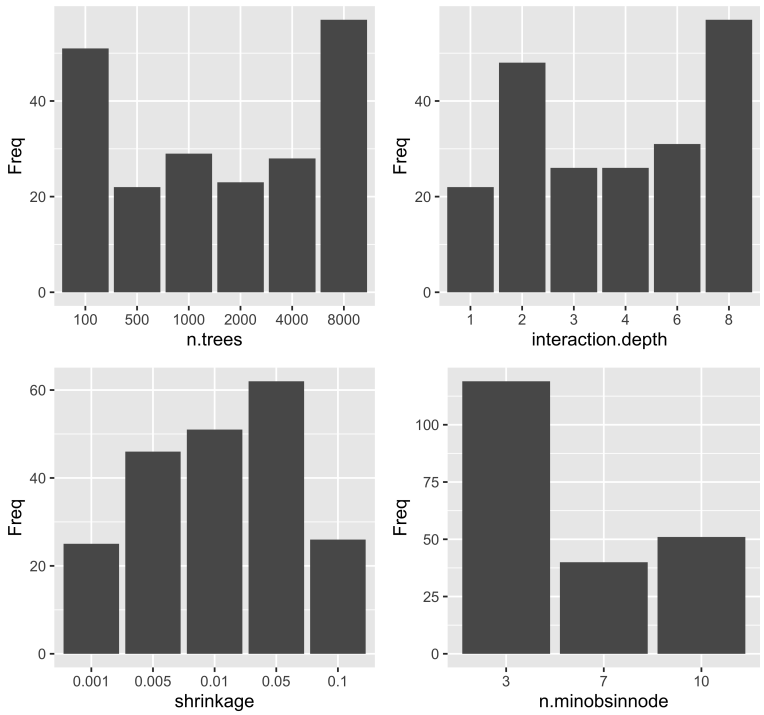

Figure E.5: Histogram of optimal hyperparameter values for GBM.

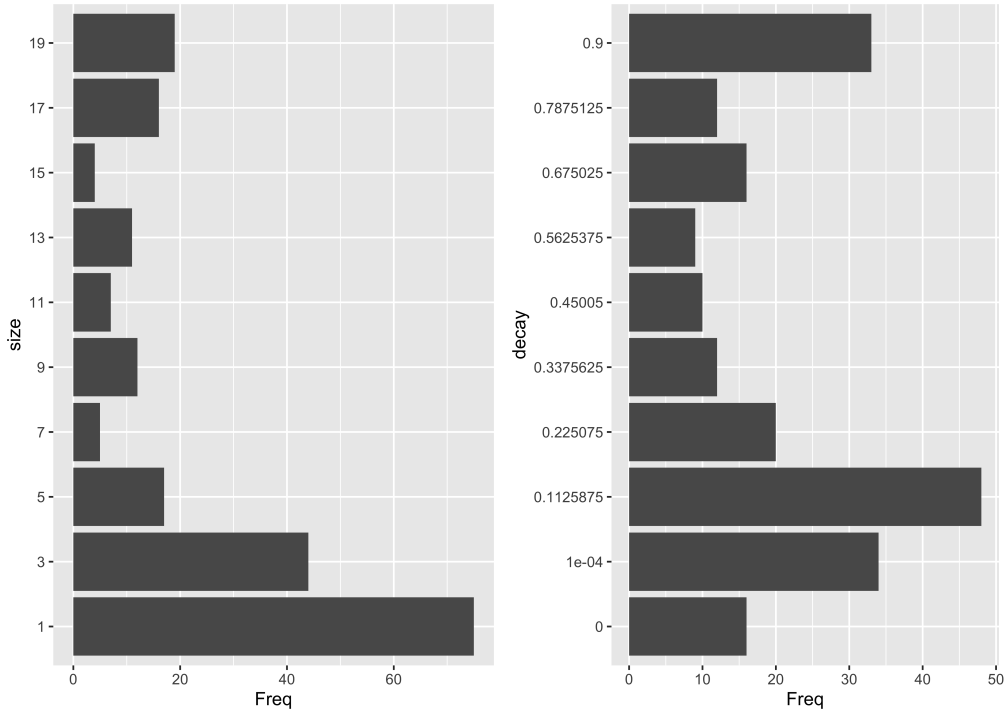

Figure E.6: Histogram of optimal hyperparameter values for ANNs.