5-6-2016

Evolutionary Weights for Random Subspace Learning

Andre Lobato Ramos
axl5988@rit.edu

R · I · T

Evolutionary Weights for Random Subspace Learning

by

André Lobato Ramos

A Thesis Submitted in Partial Fulfillment for the
Requirements for the Degree of Master of Science
in Applied Statistics

in the
School of Mathematical Sciences
College of Science

Rochester Institute of Technology
Rochester,NY
May 6, 2016

Committee Approval

Ernest Fokoue, Associate Professor, School of Mathematical Sciences Thesis Adivisor	Date
Joseph Voelkel, Professor, School of Mathematical Sciences Committee Member	Date
Steven Lalonde, Associate Professor, School of Mathematical Sciences Committee Member	Date

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."

Sherlock Holmes

Abstract

Ensemble learning is a widely used technique in Data Mining, this method allows us to aggregate models to reduce prediction error. There are many methods on how to perform model aggregation, one of them is known as Random Subspace Learning, which consists of building subspace of the feature space where we want to create our models. The task of selecting good subspaces and in turn produce good models for better prediction can be a daunting one, so we want to propose a new method to accomplish such a task. This proposed method allows for an automated data-driven way to attribute weights to variables in the feature space in order select variables that show themselves to be important in reducing the prediction error.

Keywords: Machine Learning, Ensemble Learning, Random Subspace Learning, Weighting Scheme.

Acknowledgements

I would like to begin by acknowledging CAPES and CNPq for providing my scholarship through the Brazilian Scientific Mobility Program, supporting me during my master's degree and giving me this amazing opportunity to study in the United States.

I would like to express my sincere gratitude to my advisor Dr.Fokoué for his tremendous support and encouragement, always believing that this thesis would be completed on time. For all the time he had his doors opened to help me and always finding time to help every single one of his students, even on weekends! Also, for all the good chats about Statistics, politics, soccer, music and countless other subjects.

Besides my advisor, I would like to thank Professor Voelkel and Professor Lalonde, for sharing their unmeasurable expertises beyond the classroom, and for taking the time to read this thesis and be part of my committee.

I would also like to thank John Walker at Xerox, Micheal Long and Chris Homan at RIT, for the challenges given to me during my internships and for the trust they placed on me, that I would be able to accomplish those challenges and make the most of my experience in this country.

I thank Gabriela, for all her support and help when things seemed out of control, patience when I needed it the most, for keeping me on track when I lost sight of things and without whose love I would not have been able to finish this thesis.

My friends, Lara and Vinicius for all the good times and amazing trips we made during our stay in the US.

I would like to thank my family for all the support and always believing that I could finish this master.

I also want to thank all those whom I unfairly did not mention here.

André Lobato Ramos

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In Machine Learning, the method of combining multiple models is used to reduce the prediction error; this technique is known as ensemble learning. In ensemble learning, we beging by fitting many models to the training set, each of those models is called a base learner, a base learner can be any kind of modelling techinique, and we could use different kind of base learner in the same ensemble and as we add more models to the ensemble we are saying that we are growing the ensemble. One important aspect of ensemble learning, is that we want one base learner to be different from the other, which allows for a better way to search the model space.

There are different techniques to insert variability into each base learner in the ensemble. Bryll, Gutierrez-Osuna, and Quek 2003, state that bootstrapping aggregating (bagging) (Breiman 1996a) and boosting (Freund 1995) are among the most used methods of ensemble learning, these techniques biggest target are the observations, in bagging we bootstrap the observation in order to fit each base learner, and in boosting we are also peforming bootstrap, but this time we assign higher weights to variables that have previously been missclassied by other base learners in the ensemble. The technique which will be the focus of this work is Random Subspace Method (RSM) (Ho 1998), which we want to propose a new method for performing the task of selecting subspaces.

Something we could ask ourselves when making use of the ensemble learning technique is: why use ensemble learning instead of finding a single model? An interesting answer to this question can be found in Kuncheva (2014) and it says "The realization that a complex problem can be elegantly solved using simple and manageable tools." We know that there great tools out there such as the Akaike Information Criterion (AIC) (Akaike 1974) and Bayesian Information Criterion (BIC) (Schwarz 1978) which allow us to pick good models, or it will even pick the best model in case of the latter option, if the best model is available. Those are very interesting techniques, if the researcher is trying to understand what drives the data and wants to see how the predictors change the response. If one is interested in the estimation of parameters of a model, a way to this is to minimize the risk:

$$R(\hat{\boldsymbol{\theta}}) = MSE(\hat{\boldsymbol{\theta}})$$

$$R(\hat{\boldsymbol{\theta}}) = E[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^\top (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})]$$

So given parameter space Θ we want θ such that:

$$\hat{\theta}_{\text{MSE}} = \underset{a \in \Theta}{\text{argmin}}\{R(a)\}$$

It is well known that squared loss can be decomposed into two components; bias and variance as shown below:

$$R(\hat{\theta}) = \text{Bias}^2(\hat{\theta}) + \text{Var}(\hat{\theta}),$$

The focus of ensemble learning is prediction and therefore we want to reduce the prediction error:

$$\boldsymbol{X}\hat{\theta} = \underset{a \in \Theta}{\text{argmin}}\{\text{E}[|| \boldsymbol{X}a - \boldsymbol{X}\theta ||_n^2]\}$$

As stated, model estimation is composed of bias and variance, where we have a trade-off between those two elements. This trade-off means that models with low bias will have high variance, and the opposite is also true, models with low variance have high bias. In the context of prediction, we don't necessarily want to know what the parameters of the models are or which are significant or not significant. Take for example the Spam dataset (Leisch and Dimitriadou 2010), where want to classify whether an e-mail is spam or not spam, we are not really interested in knowing what makes a spam e-mail (although some people may find this be an interesting question to be answered), all we really want to do is to predict if a new e-mail is a spam, we could find just one good model that does that, but the issue with model selection is that we select just one model, and end up throwing out other good models.So if the goal is to make correct predictions, why not also use all those other good models?

The idea is creating an ensemble of models with similar bias then averaging over them in order to reduce the variance. This idea dates back to 1979 in the work of Dasarathy and Sheela, where they combined classifiers for pattern recognition. The main point of ensemble learning is to have a lower probability of choosing a bad model for our data (Zhang and Ma 2012;2015).

As we know, there is no free lunch, so it is also important to keep in mind that this ensembling technique also brings another side, mainly when the number of models in the ensemble becomes too large. When we a large number of members of the ensemble, we require large storage space and more computational time (Baba et al. 2015).

The proposed algorithm in thesis, we want to create a way to smartly select variables from the feature space, instead of picking uniformly variables from the dataset, we want to learn which variables are important for predicting the response and which are not helping with prediction. In order to do that selection, we will start by selecting variables in the feature space, and fit the given base learner to the data then we will assess how that model

performed and use this information to give weight to the variables that were used to fit that model. The idea to do such task is following: as we grow the ensemble, we will start give higher weights to variables that prove themselves important. Thus, we will have more models in the ensemble with important variables. Besides selecting variables, this algorithm also performs bootstrap on the training data. By selecting both observations and variables, we have two sources of variation coming into the ensemble, one from the variable selection and the other coming from sampling on the sample data. The idea behind this algorithm is the same as in the theory of evolution (Darwin 1929), we want good features to survive and become more dominant in a given environment, in the case of statistics, the feature will be the feature space and the environment will be the given dataset. We will show that this algorithm is easily adapted to either regression or classification task.

In order to show how well this algorithm works, we will compare it other random subspace learning methods, by simulating data and applying it to real data and see how it performs next to the other algorithms. It will also be shown how the algorithm works empirically, by using simulated data we will show how the weights are updated and we start to see how the variables that generated the process start to stand out from the remaining features.

This thesis is divided as follows, we will begin by defining the two main models we will be using, then we will talk about ensemble learning as a whole. After that, we will describe random subspace learning, a variation of it with attribute bagging and we will also talk about a special case of a weighted version. Next, we will introduce the extension that we are proposing in this thesis, first describing the algorithm, then go into details of how the weighting scheme works. Once we are finished describing all the algorithm, we will describe how we will choose the size of the subspace for the data we will be using. Finally, we will make the comparison of all four algorithms.

Chapter 2

Methods

Random Subspace Method (RSM) consists of creating multiple models based on the feature space, rather than just a subset of the training sample. Given a feature space, the algorithm will select, uniformly, a smaller number of variables; then a model is build on that subspace of the training set. In his paper, Ho claims that this method doesn't suffer from the "curse of dimensionality", but instead, it actually takes advantage of the situation.Throughout this thesis, Random Subspace Method and Random Subspace Learning (RSSL) will be used interchangeably.

As said before, the RSM algorithm described in (Ho 1998) selects variables randomly, with a uniform distribution, with that in mind another method of selecting subspaces was proposed by Fokoué and Elshrif (2015), which as named Random Adaptive Subspace Learner (RASSEL). Their novel algorithm selects variable based on the data at hand, so for regression model they give a higher probability of selection to variables with higher correlation with the response, and for classification models, they compute a one-way ANOVA for every variable. This algorithm differs from the one proposed in this thesis in the sense that RASSEL has statics weights, whereas the one we will propose the weight will change as we grow the size of the ensemble.

In this section, we will begin by defining Multiple Linear Regression and Logistic Regression, which will be the base learners used in this thesis, then we will go over ensemble learning. After defining the above, we will talk how the ensembling of models will be done. For comparison purposes, we will first begin by explaining how the classic RSM (Ho 1998) and the adaptive version of RSM work. Afterward, we will see how RASSEL algorithm (Fokoué and Elshrif 2015) does the ensembling of learners. Finally, we will discuss the method proposed in this thesis, which we called Evolutionary Random Adaptive Subspace Ensemble Learner. Before going any further, let's define the notation that will be used in this section. This following notation will be used for all three methods. Given a dataset $\mathscr{D} = \{\boldsymbol{z_i} = (\boldsymbol{x_i}^{\mathrm{T}}, y_i)^{\top}\}$, with $\boldsymbol{x_i} = (x_{i1}, ..., x_{i1})^{\top} \in \mathcal{X} \subset \mathbb{R}^{\mathrm{p}}$ and $y_i \in \mathcal{Y}$ and we want to build models $\hat{f}$ to predict Y as a function of the predictors $\boldsymbol{X}$. All methods of ensembling will be compared by fitting a Generalized Linear Model, specifically, Multiple Linear Regression (MLR) and Logistic Regression.

2.1 Generalized Linear Model

We will be using Generalized Linear Models (GLM) (McCullagh and Nelder 1989) as the method of making models, which is a widely used method in statistics to model responses in many different scales, such as nominal, ordinal and continuous. The term response will be employed for measurements that are free to vary in response of what will be called predictors (Dobson and Barnett 2008). A GLM is in the form of:

$$g(E[Y_i \mid X_i]) = \boldsymbol{X}_i^\top \boldsymbol{\beta} \tag{2.1}$$

Where $g(.)$ is a monotonic function called link function, $\boldsymbol{X}$ is the design matrix and $\boldsymbol{\beta}$ is the set of parameters, $\boldsymbol{Y}$ is the response variable, and will be either a real number or a dichotomous variable. As mentioned above, we will not be focusing on how to estimate $\boldsymbol{\beta}$ nor making hypothesis tests to see which β_j are significant.

We will be using the identity link and logit link, which gives us Multiple Linear Regression and Logistic Regression, which are the most common methods for a continuous and binary response, respectively (Dobson and Barnett 2008).Of course, we could be using any other kind modeling techniques, such as k-Nearest Neighbors, Trees, Support Vector Machine, among others (Hastie, Tibshirani, and Friedman 2009). Let's proceed now into the some of the details of these two models.

2.1.1 Multiple Linear Regression

Given our dataset $\mathcal{D}$ with response $\boldsymbol{Y} \in \mathbb{R}^n$ and $\boldsymbol{X} \in \mathbb{R}^{n\times(p+1)}$, if we want to predict $\boldsymbol{Y}$ given $\boldsymbol{X}$ using MLR we have that:

$$\mathbb{E}[\boldsymbol{Y} \mid \boldsymbol{X}] = \boldsymbol{X}\boldsymbol{\beta}. \tag{2.2}$$

Where $\boldsymbol{Y} = (Y_1, ..., Y_n)^\top$, $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_p)^\top$. We can also say, that the response Y varies around its mean, which gives us the following equation:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{2.3}$$

Such that $\boldsymbol{\varepsilon} = (\varepsilon_1, ..., \varepsilon_n)^{\mathrm{T}}$, under the assumption that:

$$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2),$$

which leads to

$$\boldsymbol{\varepsilon} \sim N_n(0, \sigma^2 \boldsymbol{I}_n),$$

where $\boldsymbol{I}_n$ is an $n \times n$ identity matrix. Finally $\boldsymbol{X}$ is the design matrix:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1p} \\ 1 & x_{21} & x_{22} & ... & x_{2p} \\ \vdots & \vdots & \vdots & ... & \vdots \\ 1 & x_{i1} & x_{i2} & ... & x_{ip} \\ \vdots & \vdots & \vdots & ... & \vdots \\ 1 & x_{n1} & x_{n2} & ... & x_{np} \end{bmatrix}$$

with $x_{ij} \in \mathbb{R}$, for $i = 1, .., n$ and $j = 1, ..., p$.

If we want to predict the response Y of a new observation $\tilde{\boldsymbol{x}}^* = (1, x_1^*, ..., x_p^*)$ we will need to do :

$$\hat{f}(\boldsymbol{x}^*) = (\tilde{\boldsymbol{x}}^*)^\top \hat{\boldsymbol{\beta}}. \tag{2.4}$$

In order to find $\hat{\boldsymbol{\beta}}$, the estimator of $\boldsymbol{\beta}$, we want to minimize the sum of squared errors:

$$\hat{\boldsymbol{\beta}} = \underset{\beta \in \mathbb{R}^{(p+1)\times p}}{\operatorname{argmin}} \{(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})\}.$$

By using either least squares or maximum likelihood we arrive at the same solution to this problem (Neter, Wasserman, and Kutner 1989), which is:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y}. \tag{2.5}$$

It can also be shown by using the Gauss-Markov Theorem that $\hat{\boldsymbol{\beta}}$ is the Best Linear Unbiased Estimator (BLUE) of $\boldsymbol{\beta}$ (Plackett 1950), as estimation is not the focus of this thesis, this result will not be shown here. We have now seen how to predict a response in $\mathbb{R}$, now we will see how to predict a binary response.

2.1.2 Logistic Regression

For the classification task we will use logistic regression. In this thesis, the response Y of the classification problems will be a binary response. So, given the dataset $\mathscr{D} = \{(\boldsymbol{x_1}, Y_1), ..., (\boldsymbol{x_n}, Y_n)\}$ our predictors $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top \in \mathscr{D} \subseteq \mathbb{R}^p$ and the response $Y_i \in \{0, 1\}$. The design matrix $\boldsymbol{X}$ will be defined the same way as in linear regression, therefore we will have $\boldsymbol{x}_i = (1, x_1, ..., x_p)$, so let $P(Y_i = 1 \mid \boldsymbol{x}_i) = \pi_i$, from the Bernoulli distribution we have that $\pi_i = \mathbb{E}[Y_i \mid \boldsymbol{x}_i]$. The issue here, is that $\mathbb{E}[Y_i \mid \boldsymbol{X}]$ is constrained in the interval $[0, 1]$. In order to overcome that constrain, we can make use of the logit link function, which gives us:

$$\log\left[\frac{\pi_i}{1 - \pi_i}\right] = \tilde{\boldsymbol{x}}_i^\top \boldsymbol{\beta} \tag{2.6}$$

For simplicity, we will define $\tilde{\boldsymbol{x}}^\top \boldsymbol{\beta} = \eta(\boldsymbol{x}; \boldsymbol{\beta})$. In order to predict the π_i we can rewrite (2.6) as:

$$\pi_i(\boldsymbol{x}_i; \boldsymbol{\beta}) = \frac{1}{1 + e^{-\eta(\boldsymbol{x}_i; \boldsymbol{\beta})}} \tag{2.7}$$

As we have seen $\boldsymbol{Y}$ is a dicotomic variable, which means that we need to assign one of the two classes to the probability obtained Equation (2.7). In order to assign class to a given vector $\boldsymbol{x}$ we will need to construct the decision boundary as following:

$$\left\{ x \in \mathscr{X} : h(\tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}) - \tau = 0 \right\}$$

Where $h(\tilde{\boldsymbol{x}}^\top \boldsymbol{\beta})$ is defined as $P(Y_i = 1 \mid \boldsymbol{x}, \boldsymbol{\beta})$. In this work, the threshold τ will be set to $\frac{1}{2}$. In other words, we will assign the class to Y_{new} as:

$$Y_{new} = \begin{cases} 1, & \text{if } h(\tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}) > \frac{1}{2} \\ 0, & \text{Otherwise} \end{cases}$$

We have now defined MLR and Logistic Regression, in the context of ensemble learning we will be calling these models as base learners, which will be aggregated as to form the ensemble.

2.2 Ensemble Learning

Recently, in the statistical community, there has been an increasing interest in combining multiple models (Wezel and Potharst 2007). Ensemble Learning is a method that uses a collection of models and combines their prediction (Gutierrez-Osuna). What matters the most in Ensemble learning is making different errors on a given sample (Zhang and Ma 2012;2015) yielding to better and more robust predictions than a single model (Coussement and De Bock 2013). When performing ensemble learning, we can divide the problem into two steps, the first step is where we build a population of models, the second phase is when we aggregate them to make a combined prediction (Hastie, Tibshirani, and Friedman 2009).

Figure 1 shows a schematic diagram of ensemble learning. For ensemble learning to make sense, we need to inject some sort of variation from one learner to the next, and there many techniques that help us with this task. Among the most common methods to perform ensemble learning are, Bagging (Breiman 1996a), where we perform bootstrapping the observations of the training set, thus selecting a different sample to fit the model, we have random forest (Breiman 2001), which ensembles trees. Random forest tries to find the most important variables to perform the first split (the stump) and thus giving a different subspace for the models to work on. We also have boosting (Freund 1995); this algorithm also performs bootstrap on the observations but this time giving higher weights to observations that were misclassified by the previous learners. On this thesis we will be focusing on Random Subspace Method proposed by Ho in 1998 as a method to build the

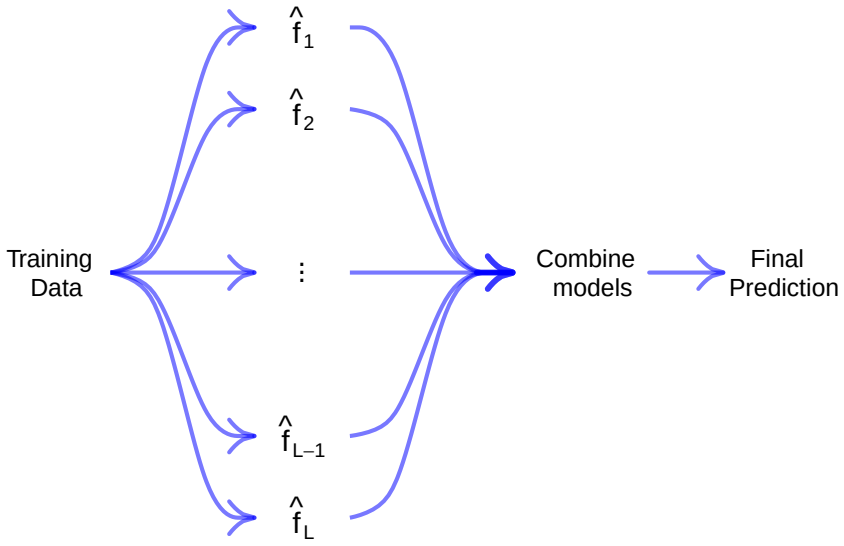

Figure 2.1: Ensmble Learning Illustration

ensemble, here the variability comes from selecting different variables to fit the model, this technique will be further discussed in section 2.3. Besides the many different strategies on how to inject variability in each of the models, we need to thing on how to assemble them to make our final prediction. In general, given the set of models $\mathcal{F}=\{\hat{f}_1, ..., \hat{f}_L\}$ and with $\boldsymbol{x}^* \in \mathcal{X}$, we can combine all of the prediction in the ensemble as:

$$\hat{f}(\boldsymbol{x}^*) = \sum_{l=1}^{L} \alpha_l \hat{f}_l(\boldsymbol{x}^*), \tag{2.8}$$

Where α_l is the weight given to the learner $f_l(.)$, this weight could be obtained while growing the ensemble such as in AdaBoosting(Freund and Schapire 1997), or obtained from a separate training set (Zhang and Ma 2012;2015). The way we will be aggregating those models is by taking the prediction of each model and averaging over them, which means $\alpha_l = \frac{1}{n}$, which gives us Equation 2.9 when performing regression:

$$\hat{f}(\boldsymbol{x}^*) = \frac{1}{L} \sum_{l=1}^{L} \hat{f}_l(\boldsymbol{x}^*); \tag{2.9}$$

When performing logistic regression the final prediction will be most frequent label:

$$\hat{f}(\boldsymbol{x}^*) = \text{argmax} \left\{ \sum_{l=1}^{L} 1_{y=\hat{f}^{(l)}(\boldsymbol{x}^*)} \right\}. \tag{2.10}$$

It is worth reinforcing the idea that this is not the only way to do this, and there are many other combination schemes (Zhang and Ma 2012;2015). The next question we should be asking is: why is perform ensemble learning instead of using variable selection? Let's have a closer look at the bias-variance tradeoff. Let $Y = f(x) + \varepsilon$, $\mathrm{E}(\varepsilon) = 0$, and $\mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2$, if we have a regression $\hat{f}(X)$, we can use the squared-loss function and find the prediction error of a given point $X = x_0$:

$$\mathrm{Error}(x_0) = \mathrm{E}[(Y - \hat{f}(x_0))^2 \mid X = x_0]$$
$$\mathrm{Error}(x_0) = \sigma_\varepsilon^2 + (\mathrm{E}[\hat{f}(x_0)] - f(x_0))^2 + \mathrm{E}[\hat{f}(x_0) - \mathrm{E}[\hat{f}(x_0)]]$$

$$\mathrm{Error}(x_0) = \sigma_\varepsilon^2 + \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\hat{f}(x_0)) \tag{2.11}$$

As we can see in Equation 2.11, the error can be partitioned into three parts under the squared loss function. The fist term is the irreducible error and can not be avoided, then we have the squared bias of the estimator of $f(x_0)$ followed by the variance of $\hat{f}(x_0)$ around the mean. Ideally, we want to have a low error, hence a low bias and variance, the issue with that is that if we have a very low bias, then our model tends to be more complex and overfits the data, which leads to high variance. On the other hand, if our model is too simple, then the variance is low but the bias is high, and the model is underfitting the data. If we are trying to make a prediction, this is a problem, and this is where ensemble learning becomes useful, because it allows to have many models with low bias (and high variance), but by averaging those low biases model we can bring the variance down as well. Another significant aspect of ensemble is that is that the general error in the ensemble is smaller than the average error of each base learner. The expression that shows this goes as follows (Krogh, Vedelsby, and others 1995); let $f(\boldsymbol{x})$ be the true function, and let $\{\hat{f}_1(\boldsymbol{x}), ..., \hat{f}_L(\boldsymbol{x})\}$ be the set of base learners with averaged prediction $\hat{f}_{avg}(\boldsymbol{x})$, and we the error $\mathrm{e}_i(\hat{f}(\boldsymbol{x}); \boldsymbol{x}) = (\hat{f}_i(\boldsymbol{x}) - f(\boldsymbol{x}))^2$, so:

$$\mathrm{e}(\hat{f}_{avg}(\boldsymbol{x}); \boldsymbol{x}) = \frac{\sum_{i=1}^{L} \mathrm{e}_i(\hat{f}(\boldsymbol{x}); \boldsymbol{x})}{L} - \frac{\sum_{i=1}^{L} (\hat{f}_i(\boldsymbol{x}) - \hat{f}_{avg}(\boldsymbol{x}))^2}{L} \tag{2.12}$$

What Equation 2.12 is telling us, is that the error of the ensemble is the average of the errors of each base learner minus the variance of the ensemble, which means that the more variability we are able to insert in the ensemble, the smaller the aggregation error will be. Of course, the expected error of aggregation cannot go to zero, as we know we are bounded by the Bayes Risk. This risk which is the infimum risk for all learners, and could only be achieved if we could have the joint distribution P_{XY}, but we can't reach this risk as we don't know the joint distribution. Let $f^*(x) = E[Y \mid X = x]$, let's call the risk $R(f^*) = R^*$, where:

$$R^* = \inf_f R(f)$$

Let $f : \mathscr{X} \to \mathscr{Y}$ be any learner, so the risk $R(f)$ is given by:

$$R(f) = E_{XY}[(f(X) - Y)^2]$$

$$R(f) = E[E[(f(X) - Y)^2 \mid X]]$$
$$R(f) = E[E[(f(X) + E[X \mid Y] + E[X \mid Y] - Y)^2 \mid X]]$$

After some manipulation, we get to:

$$R(f) = E[(f(X) - E[Y \mid X])^2] + R^* \tag{2.13}$$

Therefore, $R(f) > R^*$, showing that the risk of any technique is at least the Bayes Risk.

Now that we have seen the advantage over aggregating models, let's have a more detailed look at Random Subspace Method, or also know as Random Subspace Method.

2.3 Random Subspace Method

In Random Subspace Method, we ensemble models by selection different feature subspaces. In this scenario, the features that compose the subspace will be selected uniformly from the feature space. The RSM described in Ho 1998 is shown on algorithm 0.

Algorithm 1 Random Subspace Method

1: Select $d \lll p$ to be the size of the subspace;
2: **for** l=1,...,L **do**
3: Reform Simple Random Sample **without** replacement on Feature Space;
4: Fit model $\hat{f}_l$ on the entire training set with the chosen subspace .
5: **end for**

The final prediction is calculated as shown in equations (2.9) and (2.10).

	Y	X_1	X_2	X_3	...	X_i	...	X_p
Obs_1	y_1	x_{11}	x_{12}	x_{13}	...	x_{1i}	...	x_{1p}
Obs_2	y_2	x_{21}	x_{22}	x_{23}	...	x_{2i}	...	x_{2p}
Obs_3	y_3	x_{31}	x_{32}	x_{33}	...	x_{3i}	...	x_{3p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Obs_j	y_j	x_{j1}	x_{j2}	x_{j3}	...	x_{ji}	...	x_{jp}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Obs_n	y_n	x_{n1}	x_{n2}	x_{n3}	...	x_{ni}	...	x_{np}

Table 2.1: RSM Visualization

We can also make an adaptive version of this algorithm, so we can not only select variables, but also select observations to compose each model, just as in Bagging, proposed by Brienman (1996). Pseudocode 2 shows how to implementing the adaptive version of Random Subspace Method.

Algorithm 2 Adaptive Random Subspace Method

1: Select $d \lll p$ to be the size of the subspace;
2: **for** l=1,...,L **do**
3: Draw **without** replacement on Feature Space;
4: Draw **with** replacement observations from dataset $\mathcal{D}$ to create $\mathcal{D}^l$;
5: Fit model $\hat{f}_l$ on selected observations and subspace.
6: **end for**

	Y	X_1	X_2	X_3	...	X_i	...	X_p
Obs_1	y_1	x_{11}	x_{12}	x_{13}	...	x_{1i}	...	x_{1p}
Obs_2	y_2	x_{21}	x_{22}	x_{23}	...	x_{2i}	...	x_{2p}
Obs_3	y_3	x_{31}	x_{32}	x_{33}	...	x_{3i}	...	x_{3p}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Obs_j	y_j	x_{j1}	x_{j2}	x_{j3}	...	x_{ji}	...	x_{jp}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Obs_k	y_k	x_{k1}	x_{k2}	x_{k3}	...	x_{ki}	...	x_{kp}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Obs_n	y_n	x_{n1}	x_{n2}	x_{n3}	...	x_{ni}	...	x_{np}

Table 2.2: Adaptive Random Subspace Method

Tables 2.1 and 2.2, respectively, illustrate how RSM and Adaptive RSM work. In Table 2.1 the gray columns indicate which variables are selected for a given iteration of the RSM algorithm, and in Table 2.2 shows how the algorithm selects variables, in gray, and observations, in blue, from the dataset for a given iteration. In the next two sections 2.4 and 3, we will focus on how to select those variables, in other words, we will be giving higher weights to variables that we judge more important with the goal of selecting these variables more often, so we have better models in our ensemble.

2.4 Random Adaptive Subspace Ensemble Learner

Fokoué and Elshrif (2015) proposed Random Adaptive Subspace Ensemble Learner, which is a data-driven way to select the features from the features space. They assigned weights to the feature before beginning the ensemble and choose the subspace based on those weights. One of the ways they propose on how the weight should be computed is by calculating the correlation between the response and each of the predictors when the response is numeric, and the ANOVA F-statistic when the response is a factor. Let's now see how the RASSEL algorithm works, as described by Fokoué and Elshrif 2015. Algorithm 3 describe the process.

The final prediction for $\boldsymbol{Y}$ given the set of learners $\mathcal{F}=\{\hat{f}_1, ..., \hat{f}_\text{L}\}$ and $\boldsymbol{x}^* \in \mathcal{X}$ will be computed the same way as described in (2.9) and (2.10).

Algorithm 3 Random Adaptive Subspace Ensemble Learner

Select $d \lll p$ to be the size of the subspace;
for j=1,...,p **do**
  if $y \in \mathbb{R}^n$ **then**

$$r_j = \text{corr}(\boldsymbol{x}_j, y)$$

    Create the vector $\boldsymbol{\pi} = (\pi_1, ..., \pi_p)^{\mathrm{T}}$ as:

$$\pi_j = \frac{r_j^2}{\sum_{k=1}^{p} r_k^2}. \tag{2.14}$$

  end if
  if $y \in \{0, 1\}$ **then**

$$F_j = \frac{\text{Between groups mean squared by } x_j}{\text{Within group mean squared by } x_j}$$

    Create the vector $\boldsymbol{\pi} = (\pi_1, ..., \pi_p)^{\mathrm{T}}$ as

$$\pi_j = \frac{F_j}{\sum_{k=1}^{p} F_k}. \tag{2.15}$$

  end if
end for

$$0 \leq \pi_i \leq 1 \text{ and } \sum_{j=1}^{p} \pi_j = 1$$

for l=1,...,L **do**
  for k=1,...,d **do**
    Construct Subspace by drawing $\{j_1, ..., j_d\} \subset \{1, .., p\}$ **without** replacement from a multinomial:

$$j_k \sim Multinomial(\pi_1, ..., \pi_p);$$

  end for
  Draw **with** replacement observations from dataset $\mathcal{D}$ to create $\mathcal{D}^l$;
  Fit $\hat{f}_l$ to $\mathcal{D}^l$ in the subspace and observations selected in the previous step.
end for

Chapter 3

Evolutionary Random Adaptive Subspace Ensemble Learner

In this section, we present the algorithm that is proposed in this thesis. The intuition behind Evolutionary Random Adaptive Subspace Ensemble Learner (ERASSEL) is the same as in RASSEL, select subspaces with data-driven weights, in RASSEL this weights were based on the correlation of the response with the predictors which will lead us to have fixed throughout the building of the ensemble. With ERASSEL, we will compute those weights based on the l-th model; we will fit the model, calculate the test error for that model and assign it to the subspace that was used to fit that model. Algorithm 4 has more details on how this process works.

3.1 The Algorithm

We will begin by describing how the algorithm works on pseudocode 4, then we will go into more details on how to update the weights described in operation 10 of the algorithm in section 3.2, for regression and section 3.3 for classification.

The idea to do this weighting schemes is two-folded: (a) As we build more models, we will learn which are "good" and "bad" variables; so we can give the "bad" ones smaller probabilities to appear in the models and higher weights to "good" variables to appear more frequently. (b) It gives us the chance to see which variables are important in predicting the response in our dataset. How the updating of weights work will be discussed in more details in the next sections. ERASSEL also has the advantage that we don't need what kind of variables we have in the dataset; it will work just as well with numeric variables and with categorical variables. The final prediction of ERASSEL is calculated in the same fashion as described in (2.9) and (2.10).

Algorithm 4 Evolutionary Random Adaptive Subspace Ensemble Learner

Select $d \lll p$ to be the size of the subspace;
Initialize weight vector $\boldsymbol{\pi} = (\pi_1, ..., \pi_p)$ to uniformly select d variables;
for l=1,...,L **do**
    for k=1,...,d **do**
        Draw $\{j_1, ..., j_d\} \subset \{1, .., p\}$ **without** replacement from multinomial :

$$j_k \sim Multinomial(\pi_1, ..., \pi_p)$$

    end for
    Draw **with** replacement observations from dataset $\mathcal{D}$ to create $\mathcal{D}^l$;
    Fit model $\hat{f}_l$ on selected observations and subspace;
    Calculate error made by $\hat{f}_l$ on observations $\boldsymbol{x} \notin \mathcal{D}^l$;
    Update variable weights $\boldsymbol{\pi}$ as function of the error committed by $\hat{f}_l$
end for

3.2 Updating Weights for Regression Tasks

Let's now describe how the weights are updated in the regression scenario. We will be using the Mean Squared Error (MSE) as a way to assign weights. The MSE for model a $\hat{f}(.)$ is given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(.) - y_i)^2$$

This MSE will be calculated on observations that were not in the training set, in order words, we will be using the m Out Of Bag (OOB) observations to calculated it. We are going to call this the Out Of Bag Error (OOBE), which will be:

$$\text{OOBE} = \frac{1}{m}\sum_{i=1}^{m} I(i \in \text{OOB})(\hat{f}(\boldsymbol{x_i}) - y_i)^2 \tag{3.1}$$

So for model $\hat{f}_l(.)$ its OOBE will be stored and assigned to each of the variables that were part of $\hat{f}_l(.)$, the OOBE for model $\hat{f}_l(.)$ is:

$$\text{OOBE}_l = \frac{1}{m}\sum_{i=1}^{m} I(i \in \text{OOB})(\hat{f}_l(\boldsymbol{x_i}) - y_i)^2 \tag{3.2}$$

In order to update the weights, we want to keep track of all the OOBE's of a given variable. We want to use that information as a way to give a weight to that variable, so we will create a variable for tracking, which we will call T, and we will do the following updating scheme:

$$T^{(l)}(X_k) = \begin{cases} \frac{1}{l}\left(\sum_{i=1}^{l-1} T^{(i)}(X_k) + \text{OOBE}_l\right), & \text{if } X_k \text{was selected} \\ T^{(l-1)}(X_k), & \text{Otherwise} \end{cases}$$

Finally, we compute the weight for variable X_k for the next iteration:

$$w_k^{l+1} = \frac{\exp\left\{-T^{(l)}(X_k))\right\}}{\sum_{i=1}^{p} \exp\left\{-T^{(l)}(X_k)\right\}}. \tag{3.3}$$

As for the first round of drawing of variables, we don't have any OOBE's yet, so the approach was to assume that every variable might be important, so the all receive $\text{OOBE}_1 = 0$, hence, making all of them have weight equal to one. The advantage of doing this is that the variables that weren't selected start having higher weight as we grow the ensemble, that way having a higher probability of showing up in the future models. This agrees with the idea of ensemble learning, because as it was said, we want to insert variability in each learner of the ensemble. Of course, the weights starts to become more stable as the ensemble gets big enough.

3.3 Updating Weights for Classification Tasks

For classification tasks, we will be using the accuracy of the model to compute the weights, which is:

$$\text{ACC} = \frac{1}{n}\sum_{i=1}^{n} 1_{[Y=\hat{f}(.)]}$$

The accuracy of a model is in the interval [0,1], so the closer to 1 the more accurate is the model. Just as in regression, we will be using the m OOB observations to calculate the accuracy of model $\hat{f}_{(l)}(.)$, here we will call it the Out Of Bag Accuracy:

$$\text{OOBA} = \frac{1}{m}\sum_{i=1}^{m} I(i \in \text{OOB}) I_{[Y=\hat{f}(.)]} \tag{3.4}$$

Once more, we want to keep track of the accuracy that was obtained every time a model was fitted, so we will store the accuracy to of each of the variables that participated in the creation of model $\hat{f}_l(.)$. We will again create a variable T to keep track of what is happening in order to assign the weights, so similarly to Equation 3.3, will have:

$$T^{(l)}(X_k) = \begin{cases} \frac{1}{l}\left(\sum_{i=1}^{l-1} T^{(i)}(X_k) + \text{OOBA}_l\right), & \text{if } X_k\text{was selected} \\ T^{(l-1)}(X_k), & \text{Otherwise} \end{cases}$$

And the weight for the next round $l+1$ will simply be:

$$w_k^{l+1} = \frac{T^{(l)}(X_k)}{\sum_{i=1}^{p} T^{(l)}(X_k)}. \tag{3.5}$$

Just as we did in regression, we want to have an optimistic start, so for regression we have the first OOBE a value of zero, here we will set $\text{OOBA}_1 = 1$, and again this will give equal weights to variables in the dataset, and also this will make variables that have not yet been selected as the ensemble grow have higher weights for the next sampling of variables.

With the method on how the weights are updated, we could speculate on what will happen as the size of the ensemble grows. The first thing that comes to mind, is that this scheme will begin to give higher weights to variables that deem themselves important, as said previously. This could be thought of as a kind of variable selection, here might say that this algorithm brings the good aspects of ensemble learning, and also gives the researcher a way of understanding what variables are driving the data. Another thing we could think is, how do the final weights compare to RASSEL; could the weights that are dynamically calculated by ERASSEL converge to the statics weights of RASSEL? Let $w_e^{(l)}$ and w_r be the weights computed in ERASSEL and RASSEL respectively, and l the ensemble size, perhaps the following could happen:

$$\lim_{l \to \infty} w_e^{(l)} = w_r$$

Recall that the importance of ensemble learning is to create variability into each learner and therefore reduce the prediction error by aggregating them. So, if this convergence is in fact happening, we could argue that ERASSEL might be injecting more variability into the beginning of the ensemble, and as the ensemble grows, it begins to build more stable models as in RASSEL. Unfortunately, due to lack of time, this effect will not be theoretically explored in this work but an empirical result will be shown in the Results section.

3.4 Size of Subspace

When performing Random Subspace Learning we have a to choose a smaller number of features than contained in the original dataset, as described in algorithms 0, 2,3 and 4. Breiman (2001) suggested that the size d of the subspace should be $d = \lceil p/3 \rceil$ for regression and $d = \lceil \sqrt{p} \rceil$ for classification, but as said in Bryll, Guitierrez-Osuna and Quek (2003) the size of the subspace will change from a problem to another. So, in order to the decide the size of the subspace in the experiments made in section 5, we will first perform ERASSEL with models of size d varying in $[0, p]$ on a sample of 80% from the original dataset, calculate the error on the 20% remaining observation. If there is a size d that yields a clear minimum; otherwise we will use d where elbow of the errors happen. In case there is no clear elbow or clear minimum, we will make use of the recommendation made by Breiman (2001).

3.5 Predictive Performance

In order to compare the algorithms described above we will measure their predictive performance with the theoretical risk, which is the standard in statistical learning theory (Fokoué and Elshrif 2015) function give by:

$$\mathscr{R}(f) = \mathbb{E}[l(\boldsymbol{Y}, f(X))] = \int_{\mathscr{X}\times\mathscr{Y}} l(\boldsymbol{x}, y) dP(\boldsymbol{x}, y). \tag{3.6}$$

In Equation 3.6 we have a general formulation for risk, let's define our loss functions for both regression and classification.

3.5.1 Squared Loss

When we perform MLR, the loss function $l(\boldsymbol{Y}, f(x))$ we want to minimize is the squared loss, which is:

$$l(\boldsymbol{Y}, f(X)) = (\boldsymbol{Y} - f(X))^2 \tag{3.7}$$

Under the squared loss, we have that the best estimator of $f(X)$ is $\mathbb{E}[Y \mid X = \boldsymbol{x}]$, but this cannot be computed as we don't have the joint distribution $P(y, \boldsymbol{x})$, thus to compare the predictive performance of the algorithms we will use the average test error:

$$\text{AVTE}(\hat{f}) = \frac{1}{R}\sum_{r=1}^{R}\left\{\frac{1}{m}\sum_{t=1}^{m} l(y_{i_t}^{(r)}, \hat{f}_r(\boldsymbol{x}_{i_r}^{(r)}))\right\} \tag{3.8}$$

Where $\hat{f}_r(.)$ is the r-th model build with the data $(\boldsymbol{x}^{(r)}, y^{(r)})$. Namely, the average test error shown in (3.8) will be the mean squared error (MSE):

$$\text{MSE}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\boldsymbol{x}_i))^2 \tag{3.9}$$

It is important to notice how this function is sensitive to large distances, as we are squaring those high values, we could be using the L1 as in LASSO regression (Tibshirani 1996) function which uses the module instead of the squared function. We will now look at the Zero-One loss function.

3.5.2 Zero-One Loss

For the classification tasks, we will use the zero-one loss, which is defined as:

$$l(Y, f(X)) = 1_{[Y \neq f(x)]} \tag{3.10}$$

The average test error here will be the classification error rate which will be:

$$\text{AVTE}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} 1_{[Y \neq \hat{f}(x)]} \tag{3.11}$$

We have defined both loss functions for regression and classification, so now we have a way to assess the performance of each of the random subspace learning algorithm that we will be using, thus giving us the means to compare among them. We can now look at the data on which we will testing these techniques.

Chapter 4

Data

As a way to compare all the algorithms we have described, we will test them in different situations. First, we will be using simulated data in two distinct scenarios, one with orthogonal data and another data that has multicollinearity; then we will use real word data. Both, simulated and real world data, will be described below.

4.1 Simulated Data

In order to compare the algorithm, they will be first tested by using simulated data that way we have more control over what is happening in the data. We will simulate data for that is suitable to perform regression and data for logistic regression.

4.1.1 Simulated Data for Multiple Linear Regression

We will simulate the data to perform regression analysis as following:

$$\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

,

where $\mu \in \mathbb{R}^p$ and Σ a $p \times p$ positive semi-definite matrix. We have that the Multivariate Normal is denoted by:

$$\phi(\boldsymbol{x}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^p \mid \Sigma \mid}} \exp\left\{\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right\}. \tag{4.1}$$

For this simulation we will have that the true, but unknown, function that is generating the response is given by:

$$Y = 1 + x_1 + x_4 + x_5 + \varepsilon$$

,
with $\varepsilon \sim N(0, 4)$. We will create three hundred observation and ten variables using this method.

4.1.2 Simulated Data for Logistic Regression

In a similar way as described in section 4.1.1, we will use 4.1 to generate the variable. The difference is that now we will need to have a dictonomic response Y. To achieve this, we will begin by creating Y as a function of X_1, X_4 and X_5:

$$\eta(X) = 1 + X_1 + X_4 + X_5,$$

Now, we will use the logit transformation on the response:

$$\pi_i = \frac{1}{1 + \exp\{-\eta(\boldsymbol{x_i}; \boldsymbol{\beta})\}}$$

This will return values for π_i in the interval [0,1]. Finally, in order to have the binary response we will simply do the following then in order to assign a class to Y we will sample from a Bernoulli:

$$Y_i \sim \text{Bern}(\pi_i)$$

Just as in the simulation for regression, we will be simulating 300 observations with 10 variables.

4.1.3 Real Data

Besides the simulated data, we also want to see how the algorithms perform with real data. We will use data that is available on statistical software R (R Core Team 2015) and libraries MASS (Venables and Ripley 2002), mlbench (Leisch and Dimitriadou 2010), kernlab (Karatzoglou et al. 2004) and ggplot2(Wickham 2009). We will also make use of the prostate cancer dataset (Stamey et al. 1989), which has a situation where $p \ggg n$, which means that we would have a lot of trouble fitting a regular linear model here, and thus RSSL becomes very useful in this situation. Table 4.1 has more information on the datasets that we will be used, such as dimension, the task that will be performed and the response variable[1] of each dataset.

[1]Some responses were transformed in order to have a more symmetric distribution and stay within the assumptions of MLR.

Dataset	Observations	Variables	Response	Task
Boston	506	14	Per capita crime rate[1]	Regression
Diamonds	53940	7	Weight of diamond	Regression
Cars93[2]	93	16	Car Price[1]	Regression
Spam	4601	58	E-mail is spam or not spam	Classification
Pima	532	8	Pima Indian women with diabetes or not	Classification
Ionosphere	531	35	Good or bad structure in the ionosphere	Classification
Prostate	79	501	Patient has prostate cancer or not	Classification

Table 4.1: Data Sets

4.2 Implementation

Every algorithm that were mentioned in sections 2.3, 2.4 and 3, the simulated data mentioned in sections 4.1.1 and 4.1.2, as well as all the analysis that will be discussed in section 5, will be implemented by using the statistical package R (R Core Team 2015). Selected code is available In appendix A.

Chapter 5

Experiments and Results

The comparative results for the simulated data will be presented in this section, both in regression and logistic regression. Firstly, we will show the results from the simulated data, along with other features of the algorithm. Then we will show all four algorithms described in sections 2.3, 2.4 and 3 compare to each other. For each of the comparisons, we will split the dataset into training and test sets, where 70% of the data will be used to as the training set, when performing classification we want both classes to appear to on both training and test sets, so we will perform a stratified sample on both labels with 70% each label on the training set and 30% on the test set. When the splitting of the data is done, we will fit the models by using each of the algorithms to the training set and calculate the error on the test. The process of splitting the data, fitting models and calculating the errors will be replicated fifty times. Each of the ensemble will be composed of two hundred models, besides that, as described in algorithm 4, ERASSEL computes the out of bag error, so the training set will again be split into two where 30% will be used in order to calculate the out of bag error.

5.1 ERASSEL at Work

Before comparing the algorithm, let's we take this opportunity to show how the algorithm for ERASSEL is working under the hood. To achieve that, we will begin by showing how the weight changes over time and the final weights given to every variable in the feature space; we will also show how it performs when there are highly correlated variables in the dataset.

First, let's see how ERASSEL performs with the simulated data and if the algorithm is able to pick the actual variables that are generating the process. So, for this simulated problem, we will run two scenarios, one where the number of variables in the models in the ensemble is exactly the number of variables of the true function, and another one where we will set the number of variables to a larger number, for this example, we will use five variables. This means that we should see ERASSEL assign higher weights to X_1, X_4 and X_5 than the remaining variables seven variables, because of how we simulated this data as explained in sections 4.1.1 and 4.1.2.

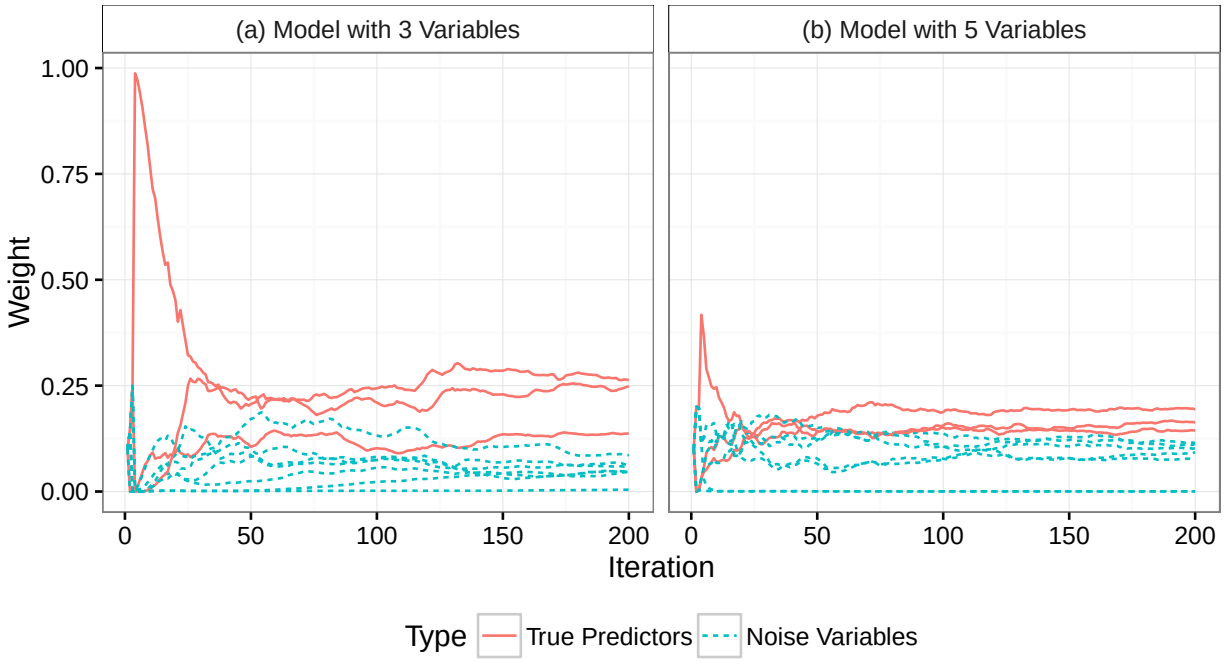

Figure 5.1: Variable Weight Path - Orthogonal Data

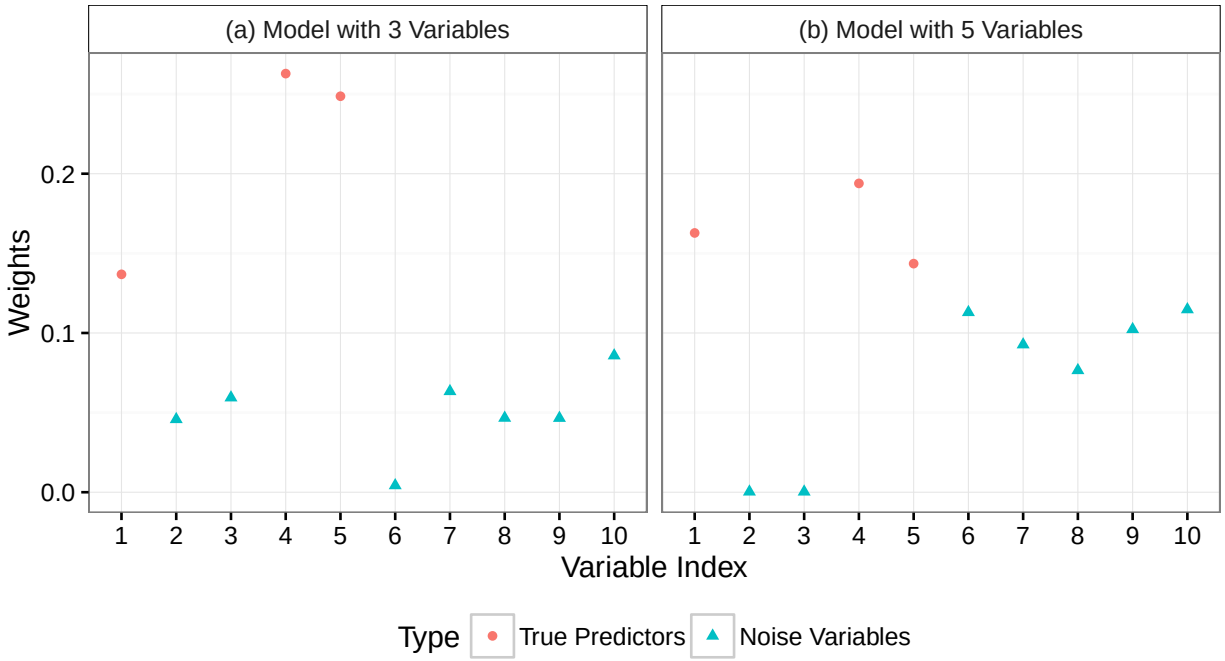

Figure 5.2: Final Variable Weight - Orthogonal Data

As we can see, Figure 3 shows that in both cases, with three and five variables, X_1, X_4 and X_5 were given the three highest weights. As mentioned in section 2.4, RASSEL assign *a priori* weights to each feature, here in this weight based on the features correlation with the response variables. Let's see how the fixed weights from RASSEL compare to ERASSEL's dynamic approach.

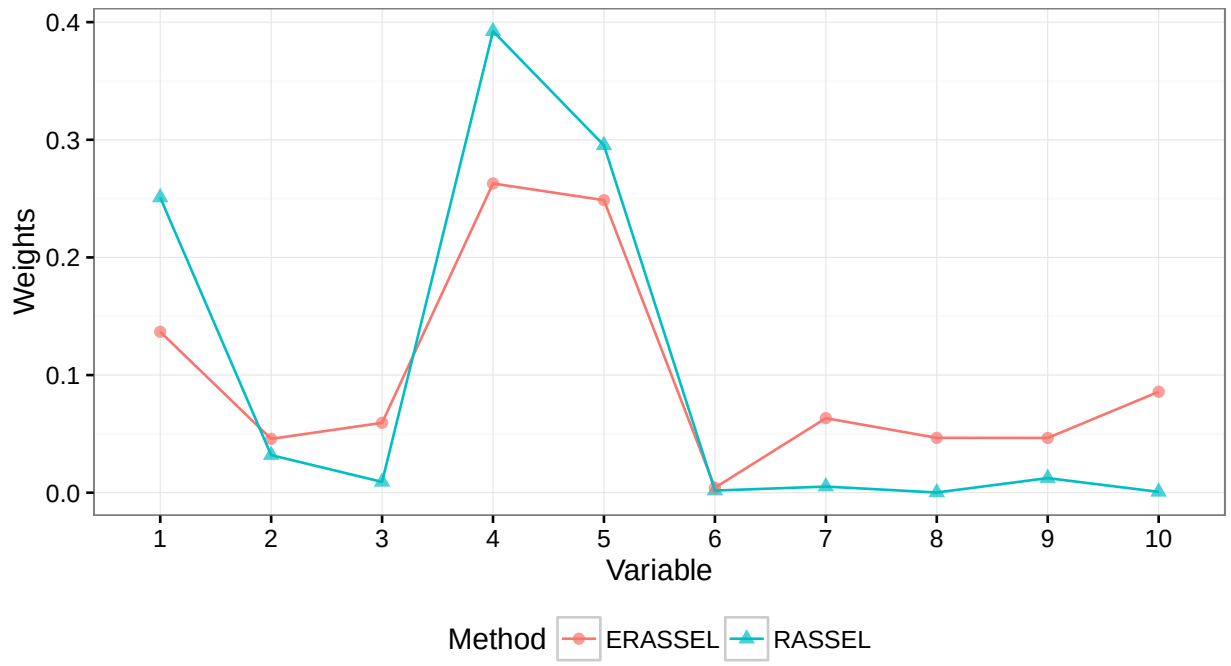

Figure 5.3: Final Variable Weight - Final Weights Comparison between ERASSEL and RASSEL

Figure 4 shows that for this simulation, these weights have the same pattern, with the important variables X_1, X_4 and X_5 having the highest weights, in fact, they are even in the same order relative to one another, i.e., in both algorithm X_4 was assigned the highest weights, followed by X_5 then X_1.

We have seen that ERASSEL can pick up the variables that are generating the response. Let's see what happens when we have highly correlated data. In order to this, we will make X_7 a linear function of X_1 and introduce some noise. This is the linear function we will use:

$$X_7 = 2X_1 + \eta$$

where $\eta \sim N(0, 1)$. Let $\hat{\boldsymbol{\rho}}$ be the correlation matrix of the simulated data obtained after performing this transformation on X_7. As we can see the correlation between X_1 and X_7 is high; give our sample it was estimated to be 0.87.

$$\hat{\boldsymbol{\rho}} = \begin{bmatrix} 1.00 & 0.45 & -0.32 & 0.19 & -0.35 & -0.03 & 0.87 & 0.33 & 0.38 & -0.09 \\ 0.45 & 1.00 & 0.29 & -0.04 & -0.06 & -0.40 & 0.30 & 0.41 & 0.44 & 0.44 \\ -0.32 & 0.29 & 1.00 & 0.30 & -0.42 & -0.38 & -0.29 & 0.15 & 0.34 & 0.30 \\ 0.19 & -0.04 & 0.30 & 1.00 & -0.54 & -0.06 & 0.35 & -0.08 & 0.05 & -0.50 \\ -0.35 & -0.06 & -0.42 & -0.54 & 1.00 & -0.26 & -0.34 & 0.02 & -0.69 & -0.17 \\ -0.03 & -0.40 & -0.38 & -0.06 & -0.26 & 1.00 & 0.13 & -0.38 & 0.25 & 0.05 \\ 0.87 & 0.30 & -0.29 & 0.35 & -0.34 & 0.13 & 1.00 & 0.13 & 0.41 & -0.14 \\ 0.33 & 0.41 & 0.15 & -0.08 & 0.02 & -0.38 & 0.13 & 1.00 & 0.29 & 0.02 \\ 0.38 & 0.44 & 0.34 & 0.05 & -0.69 & 0.25 & 0.41 & 0.29 & 1.00 & 0.67 \\ -0.09 & 0.44 & 0.30 & -0.50 & -0.17 & 0.05 & -0.14 & 0.02 & 0.67 & 1.00 \end{bmatrix}$$

Let's see how this will affect ERASSEL.

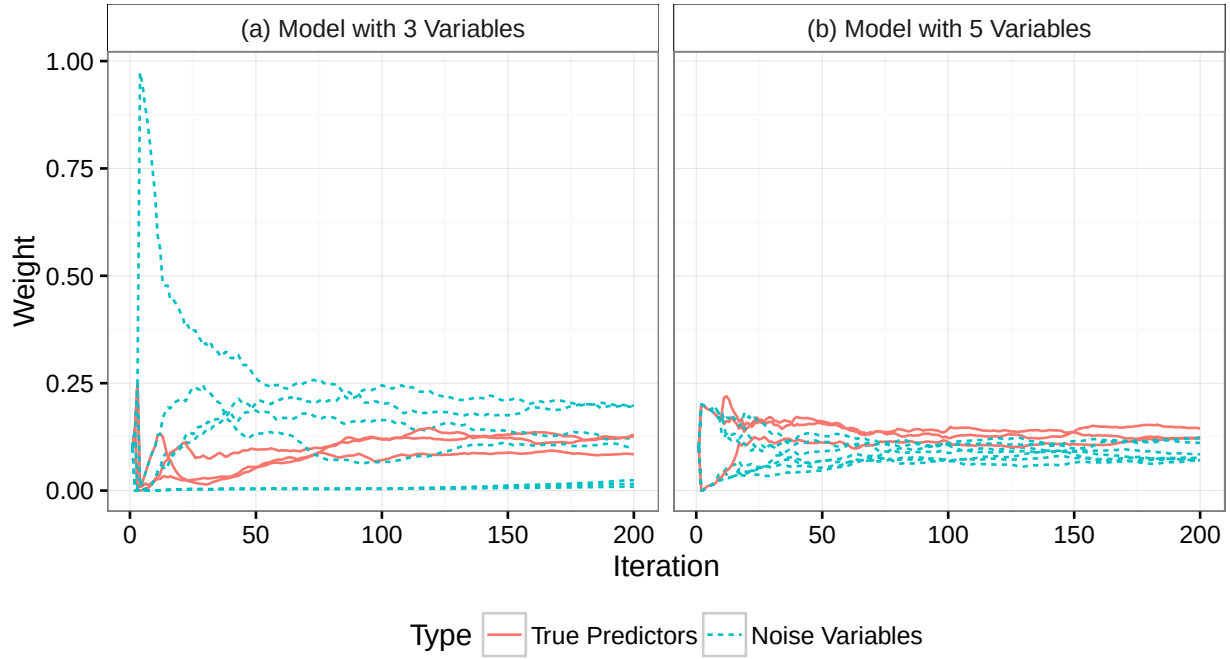

Figure 5.4: Variable Weight Path - With Multicolinearity

In Figure 5 and Figure 6 we see that this high correlation is affecting ERASSEL negatively, as the algorithm is no longer able to pick X_1, X_4 and X_5 from the remaining variables.

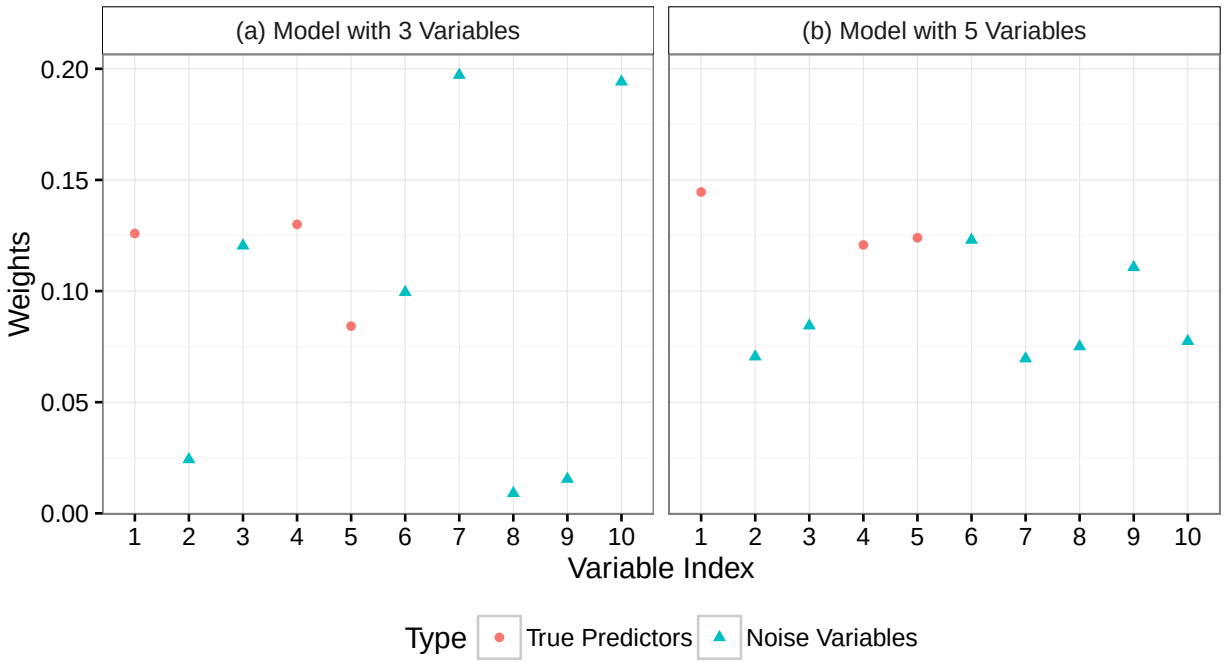

Figure 5.5: Variable Weights - With Multicolinearity

5.2 Comparative Results

Now that we have seen how ERASEEL works, we will compare how the four algorithms described previously compare with one another. As mentioned before, we will compare their respective errors, and see which of the algorithms yields the best results.

5.3 Regression

Let's begin by comparing how they perform when aggregating regression models. We will begin by the simulated data with no correlation between variables, then we will move on to real data. We will begin by choosing the size of the subspace then proceed to fitting the models and calculating their errors, these last two steps will be done 50 times as mentioned before. Also as a reminder, the original dataset will be split into training, with 70% of the data and test containing the remaining 30%.

As defined in section 3.4, we will select the size of the subspace as the size that yields the smallest MSE by fitting RASSEL to sample of the training data. In Figure 7 we see that the error decreases until we reach 5 variables in the subspace, then it begins to increase again, so we will select 5 as the size for our models.

For the simulated regression, we can see that RASSEL and ERASSEL are the two best algorithms among the 4, as we can see in Table 5.1 RASSEL yields the smallest mean error, while ERASSEL has the smallest median error.

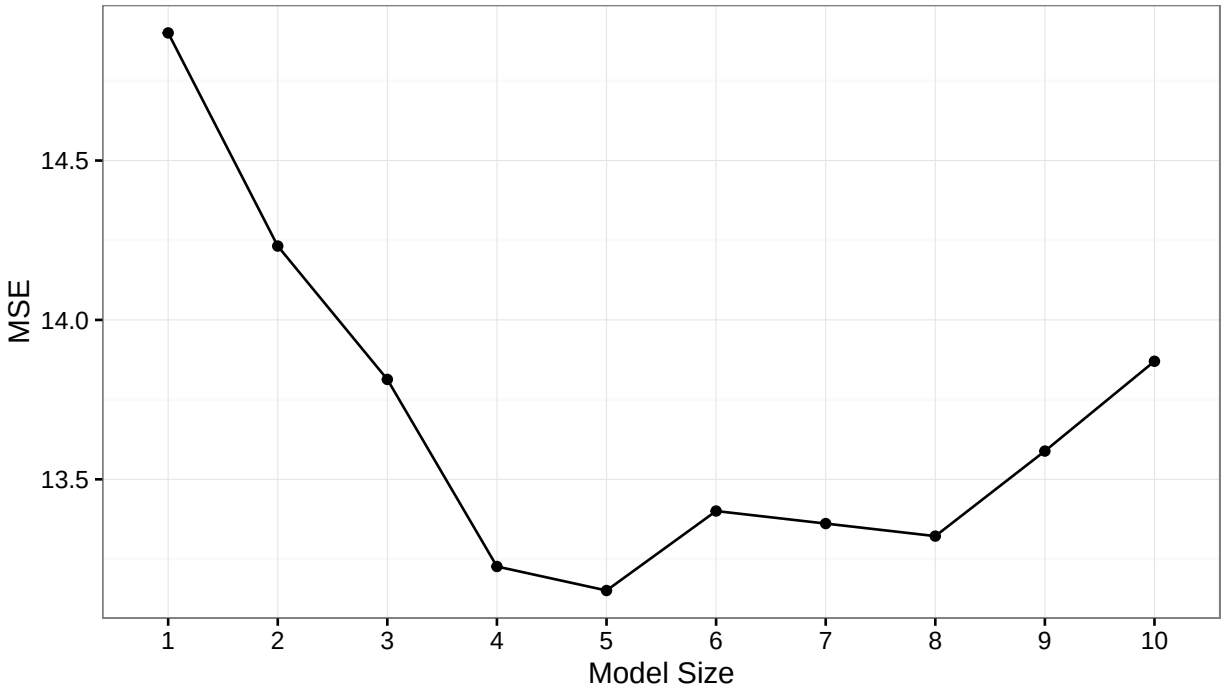

Figure 5.6: Best Model Size - Simulated Regression

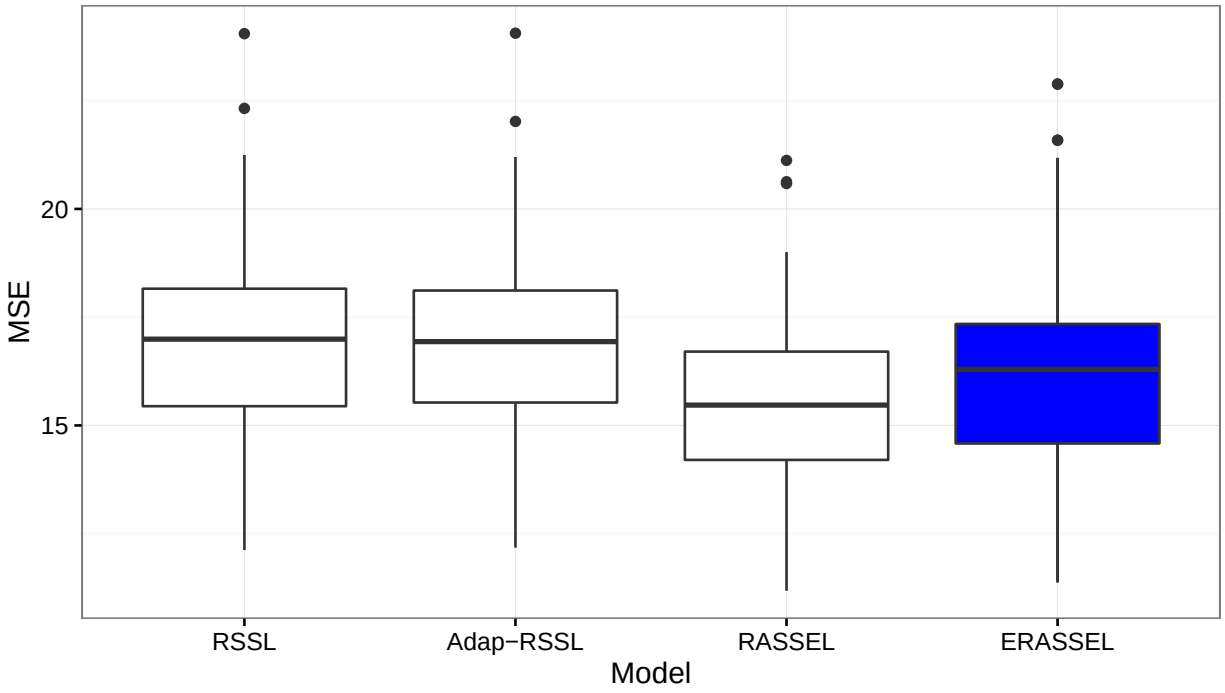

Figure 5.7: Boxplot of Errors by Model - Simulated Regression

	Model	Mean	Median	Var
1	RSSL	17.00	16.99	5.73
2	Adap-RSSL	16.98	16.94	5.59
3	RASSEL	15.52	15.47	4.86
4	ERASSEL	16.17	16.30	6.11

Table 5.1: MSE Summary by Model - Simulated Regression Data

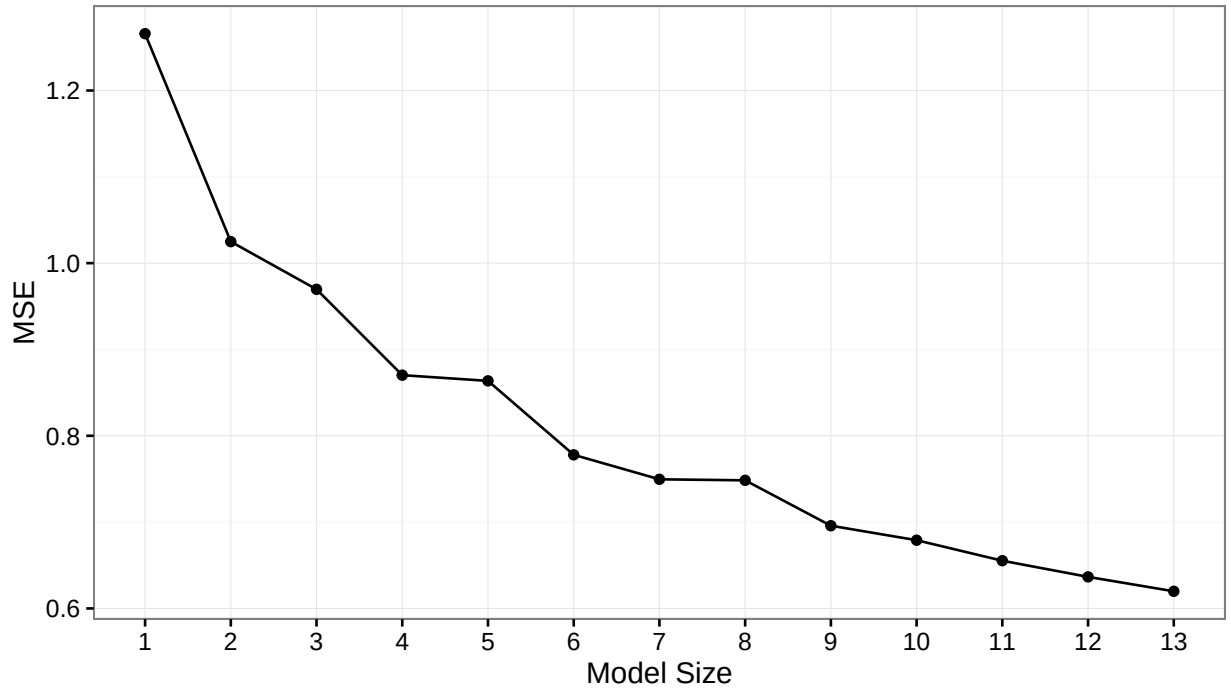

Figure 5.8: Best Model Size - Boston Dataset

Figure 9 shows that it very hard to pick the size of the subspace, as every variable we add the prediction improves, but let's pick 7 as the number of variable for our models, since it is a little over half of the total number of variable in the dataset.

Looking at 5.2 and Figure 10 we see that just as with the simulated data, the best ensemble algorithm is RASSEL, followed by ERASSEL. It is interesting to see that they all minuscule variance.
Now let's move to the diamonds dataset. The MSEs obtained for this data set were to the order of 10^{-3}, so for better displaying the results on the table, the MSES were multiplied by 1000.

	Model	Mean	Median	Var
1	RSSL	0.69	0.69	0.01
2	Adap-RSSL	0.69	0.69	0.01
3	RASSEL	0.64	0.64	0.01
4	ERASSEL	0.67	0.67	0.01

Table 5.2: MSE Summary by Model - Boston Dataset

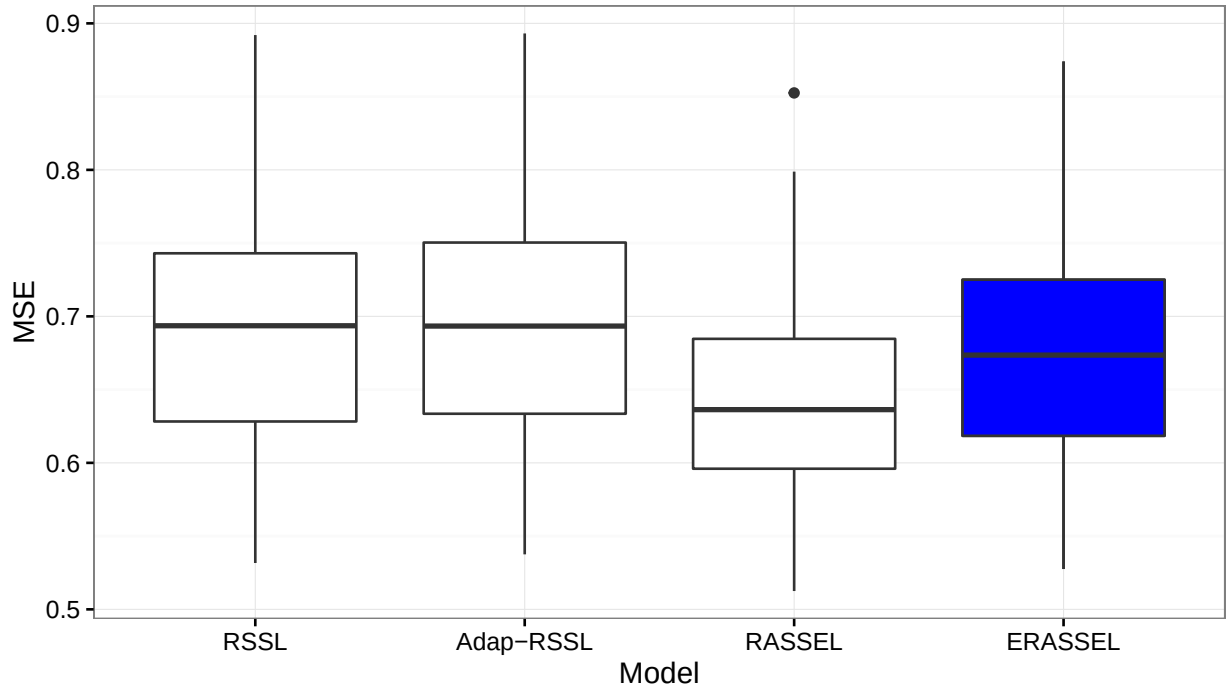

Figure 5.9: Boxplot of Errors by Model - Boston Dataset

It seems that three variables per model would be a good size with the diamonds dataset. The results are shown in Table 5.3 and Figure 12 we can see that again RASSEL was the best algorithm, followed by ERASSEL and Adaptive RSSL the had similar performance. We will now go to our last experiment with regression using the car dataset. Just like the Diamonds dataset, we obtained very small MSE and so we multiplied them by 1000.

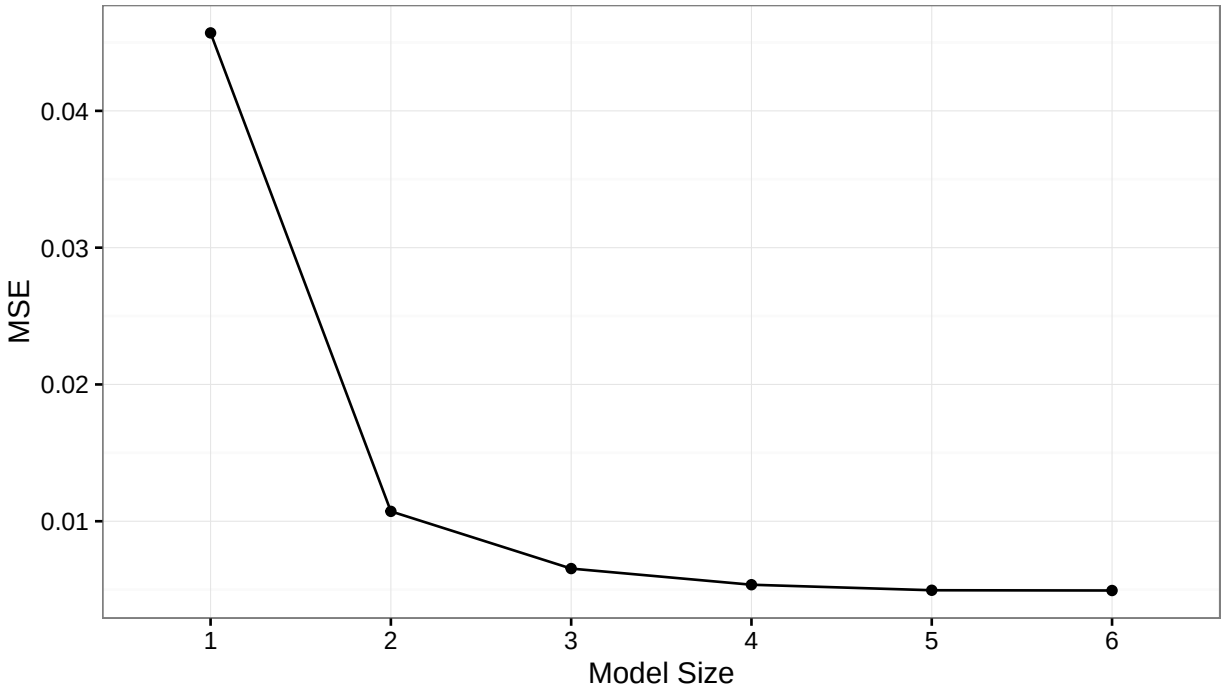

Figure 5.10: Best Model Size - Diamond Dataset

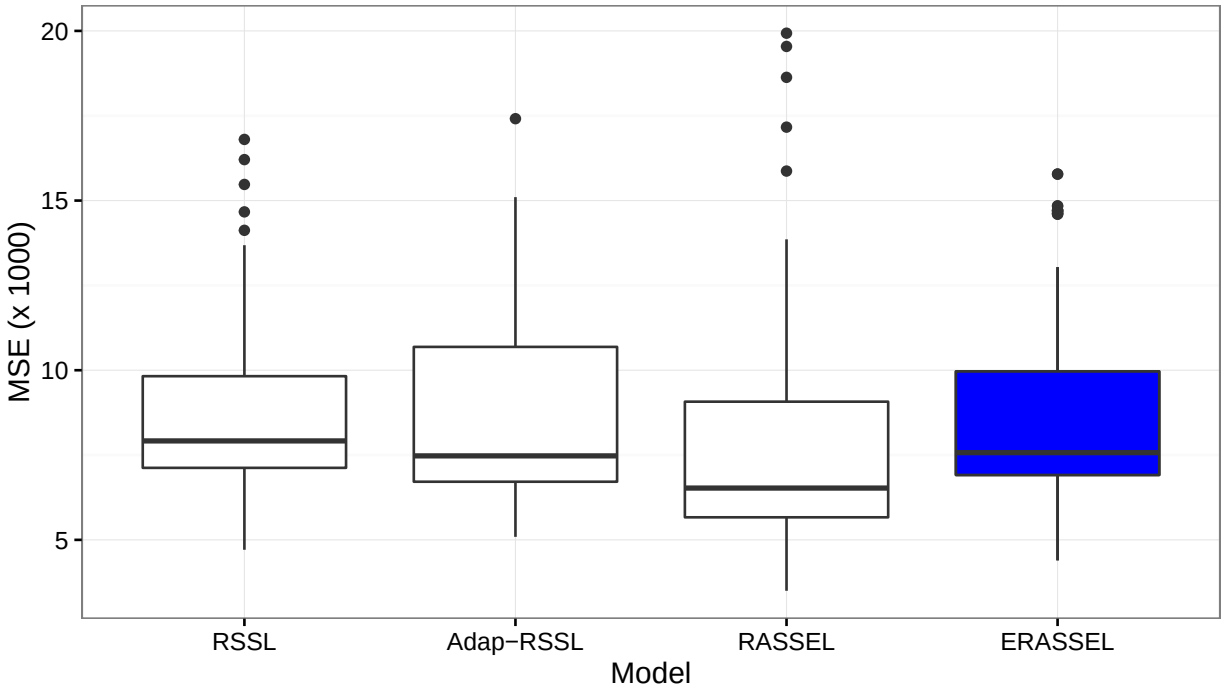

Figure 5.11: Boxplot of Errors by Model - Boston Dataset

	Model	Mean	Median	Var
1	RSSL	8.88	7.92	7.69
2	Adap-RSSL	8.76	7.48	8.51
3	RASSEL	8.13	6.53	16.45
4	ERASSEL	8.74	7.57	8.42

Table 5.3: MSE(x 1000) Summary by Model - Diamonds Dataset

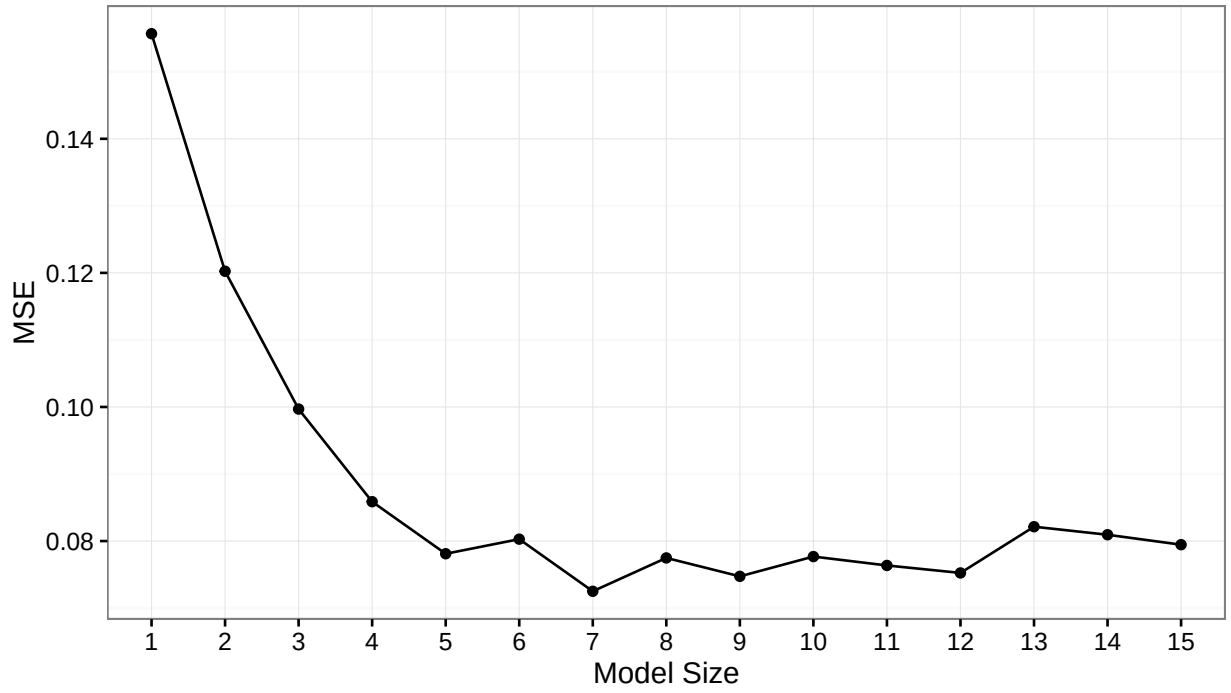

Figure 5.12: Best Model Size - Car93 Dataset

Based on Figure 13 we picked seven as the number of variables in the model.

We can see that for this dataset RASSEL didn't perform as well as before, in fact plain RSSL was the best algorithm. With this we have finished the experiments for regression, as we saw, mostly RASSEL performed slightly better than the others, but the differences in the results aren't that significant as we can see the means are very close to each other; ERASSEL seems to be a viable algorithm to perform ensemble learning. We will now see how these algorithms perform with logistic regression.

	Model	Mean	Median	Var
1	RSSL	57.29	57.16	344.15
2	Adap-RSSL	57.60	57.50	357.73
3	RASSEL	58.47	56.71	349.55
4	ERASSEL	57.79	59.17	347.89

Table 5.4: MSE(x 1000) Summary by Model - Cars93 Dataset

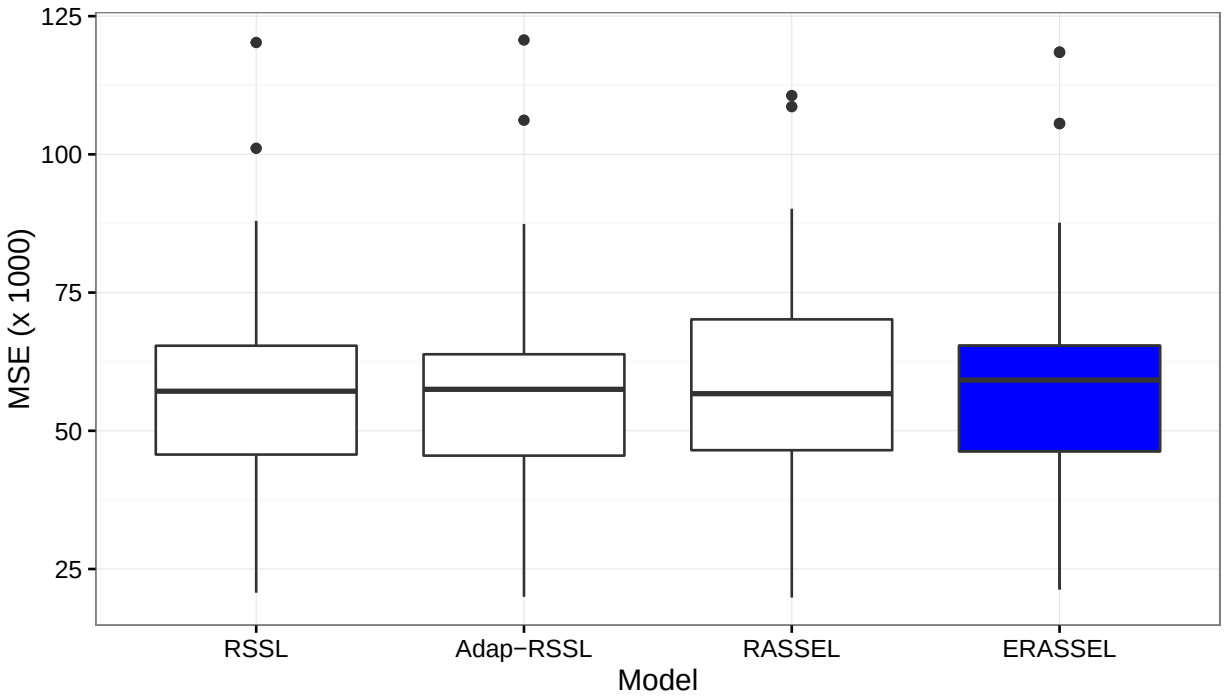

Figure 5.13: Boxplot of Errors by Model - Boston Dataset

5.4 Classification

Now, let's see how those algorithms compare when performing classification tasks. Just as we did for regression, let's begin by looking at the results for the simulated data and then real data. We will also follow the same steps as we did for regression - choose the size of subspace, fit models and calculate the error.

It seems that after models of size seven the error stabilizes and does not decrease that much. So let's use seven as the size of our models for this problem.

By looking at Figure 16 and Table 5.5 we can see that RSSL and RASSEL are performing better than the other two ensemble methods on the predicting the simulated data. Next, we can see how the methods compare with the real world data.

Based on Figure 17 it seems that 33 will be a good model size for the spam dataset.

The Spam dataset was exhaustively analyzed in Fokoué and Ly (2014), they found, empirically, that the best accuracy a classifier could achieve was 96.40%, in other orders, a

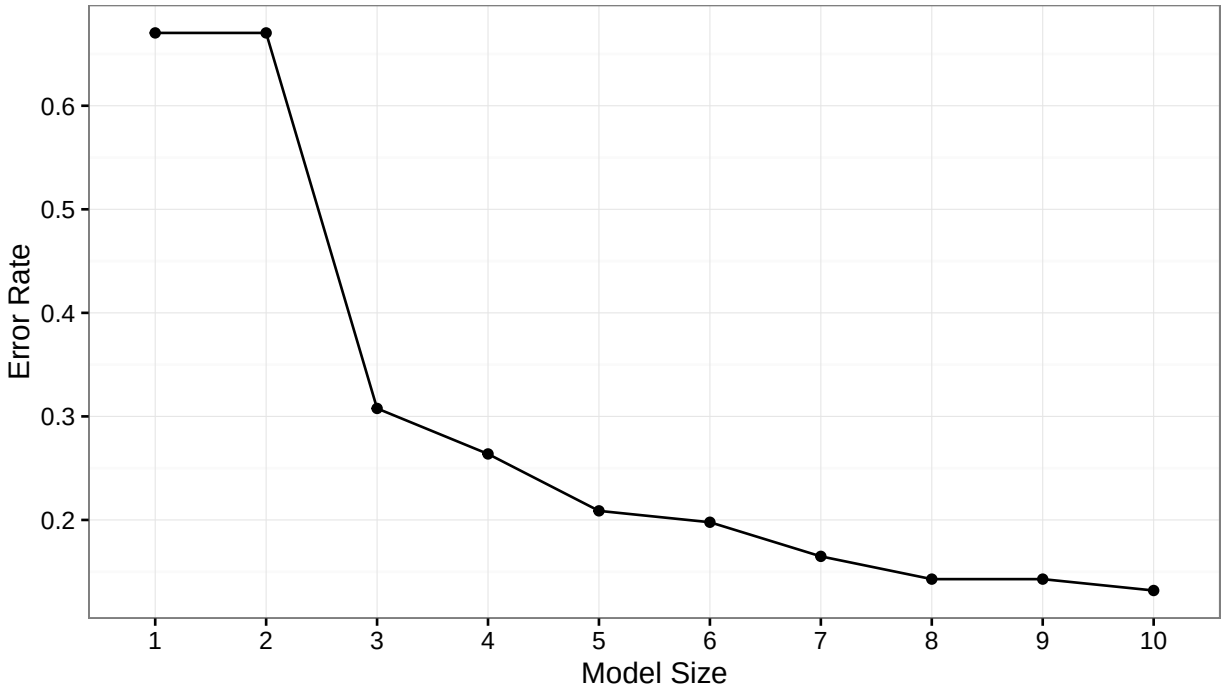

Figure 5.14: Best Model Size - Simulated Logistic Regression

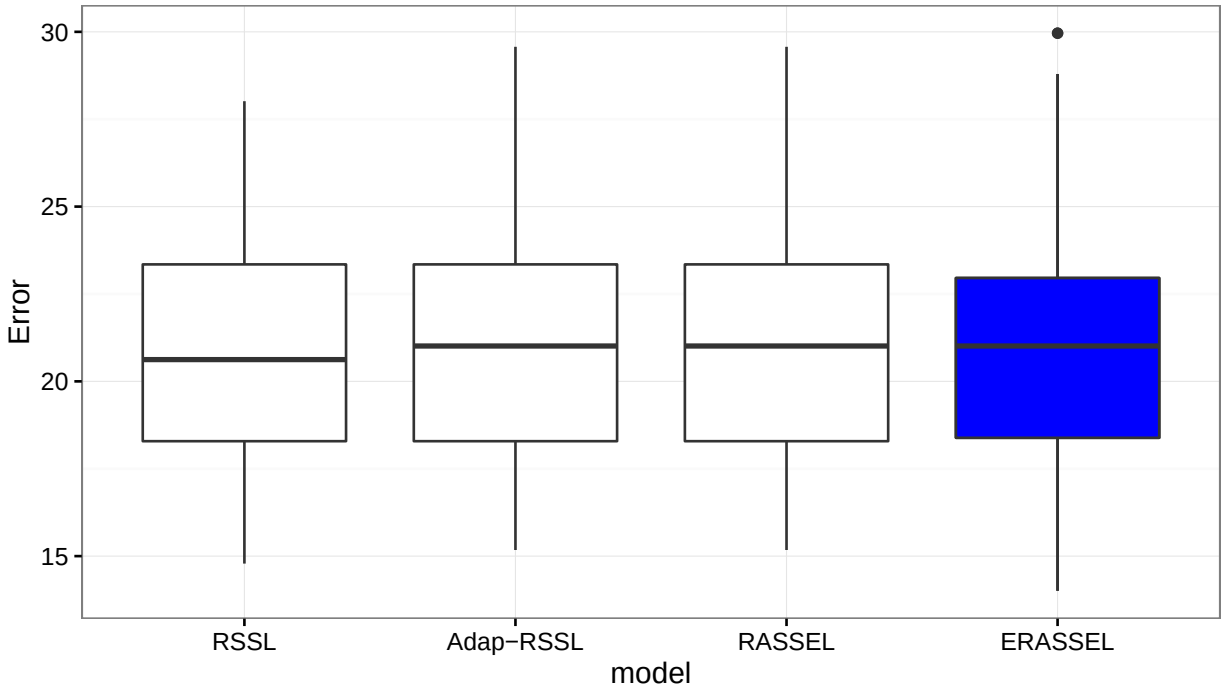

Figure 5.15: Boxplot of Errors by Model - Simulated Logistic Regression

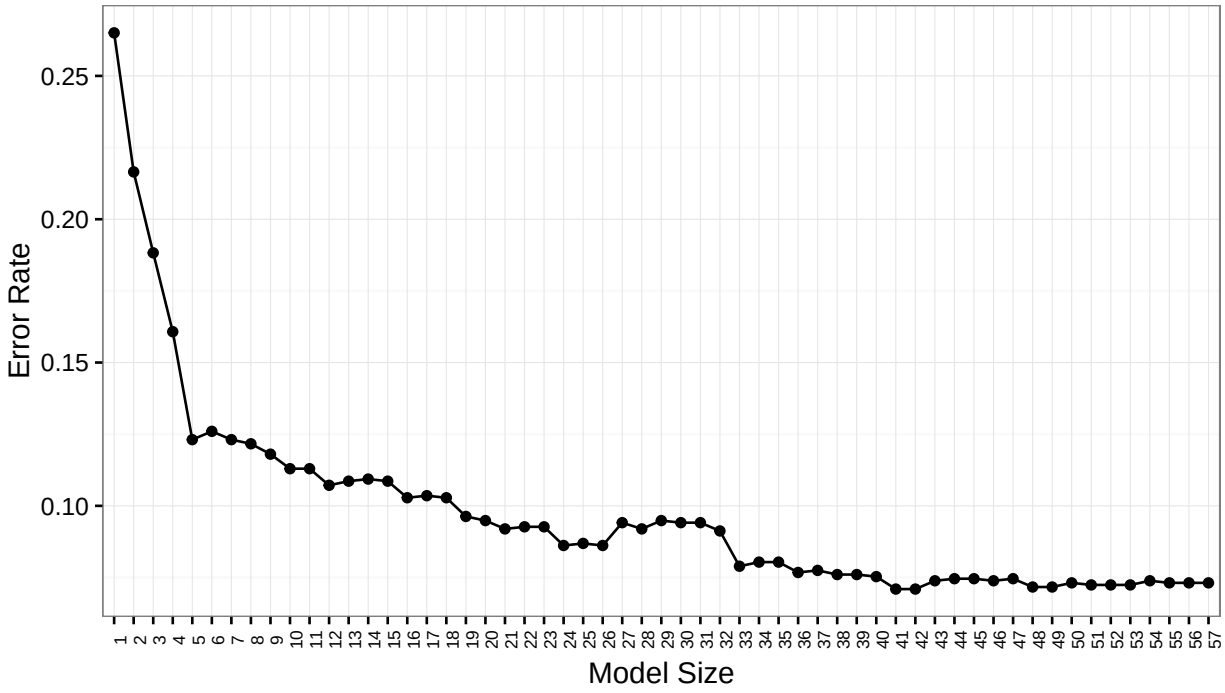

Figure 5.16: Best Model Size - Spam Dataset

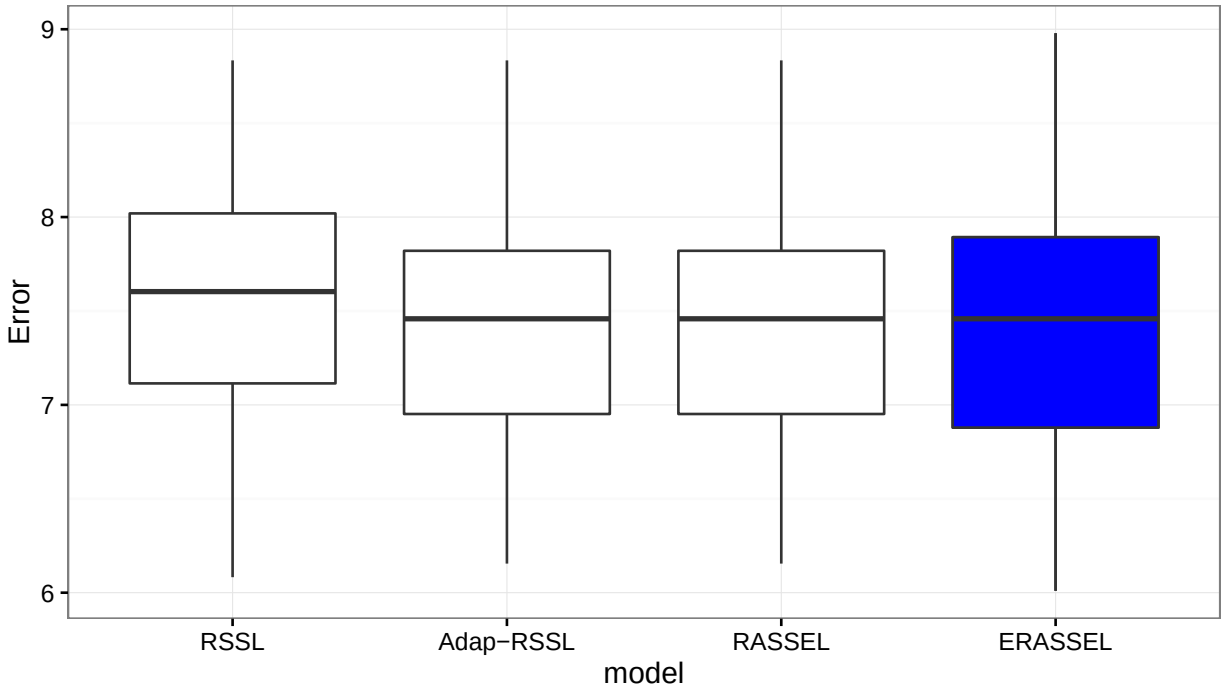

Figure 5.17: Boxplot of Errors by Model - Spam Dataset

	Model	Mean	Median	Var
1	RSSL	20.82	20.62	11.38
2	Adap-RSSL	21.29	21.01	11.90
3	RASSEL	21.29	21.01	11.90
4	ERASSEL	21.04	21.01	11.93

Table 5.5: Error Rate Summary by Model - Simulated Logistic Regression

	Model	Mean	Median	Var
1	RSSL	7.56	7.60	0.44
2	Adap-RSSL	7.42	7.46	0.36
3	RASSEL	7.42	7.46	0.36
4	ERASSEL	7.38	7.46	0.44

Table 5.6: Error Rate Summary by Model - Spam Data

3.6% error rate by using Ensembled Decision Trees. Here we have that ERASSEL had the smallest error rate out of the four methods compared, at 7.38%, RSSL was the one that had the worst median, the results are found in Table 5.6 and Figure 18.

Looking at Figure 19 we see that the minimum error occurs when we have seven variables in the model. Obviously, it wouldn't make sense to use RSM if we used every variable in the dataset, so the next best value to select as the model size seems to be three variables, where it stabilizes.

	Model	Mean	Median	Var
1	RSSL	22.48	22.36	5.21
2	Adap-RSSL	22.65	22.98	5.31
3	RASSEL	22.65	22.98	5.31
4	ERASSEL	22.67	22.36	5.72

Table 5.7: Error Rate Summary by Model - Pima Data

In Table 5.7 we see that although ERASSEL has the highest average error rate it also has the smallest median error rate, and by looking at Figure 20 we can see that it is very similar distribution to RASSEL and adaptive RSSL.

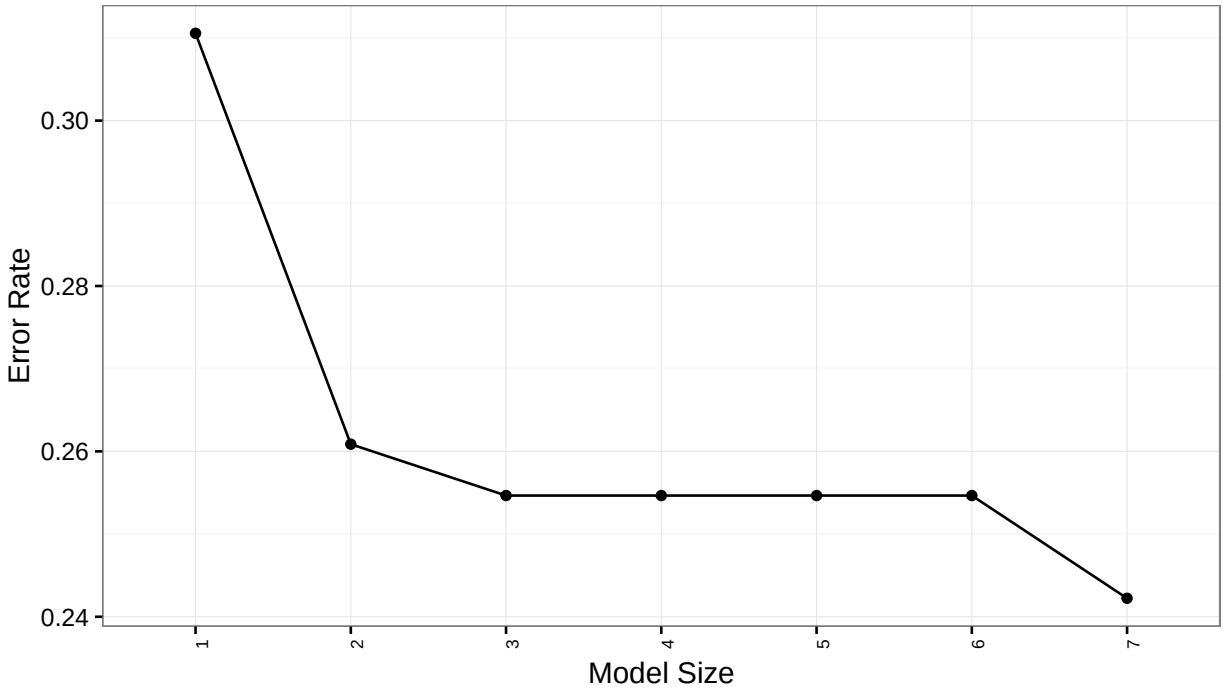

Figure 5.18: Best Model Size - Pima Dataset

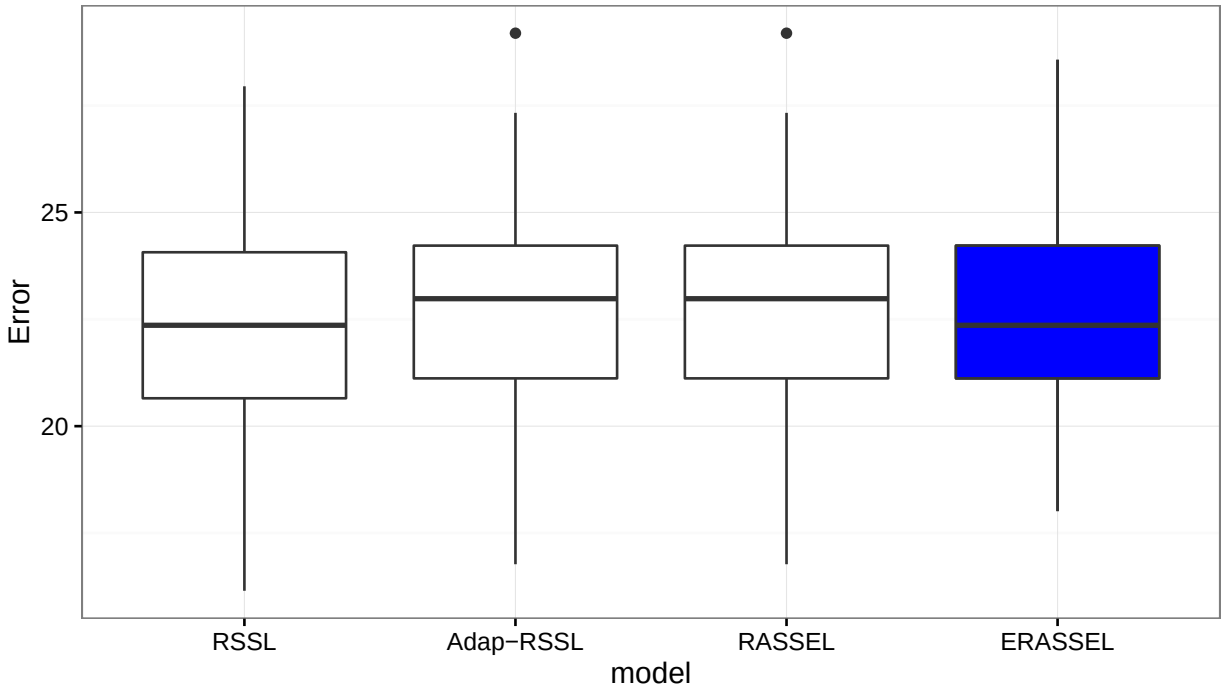

Figure 5.19: Boxplot of Errors by Model - Pima Dataset

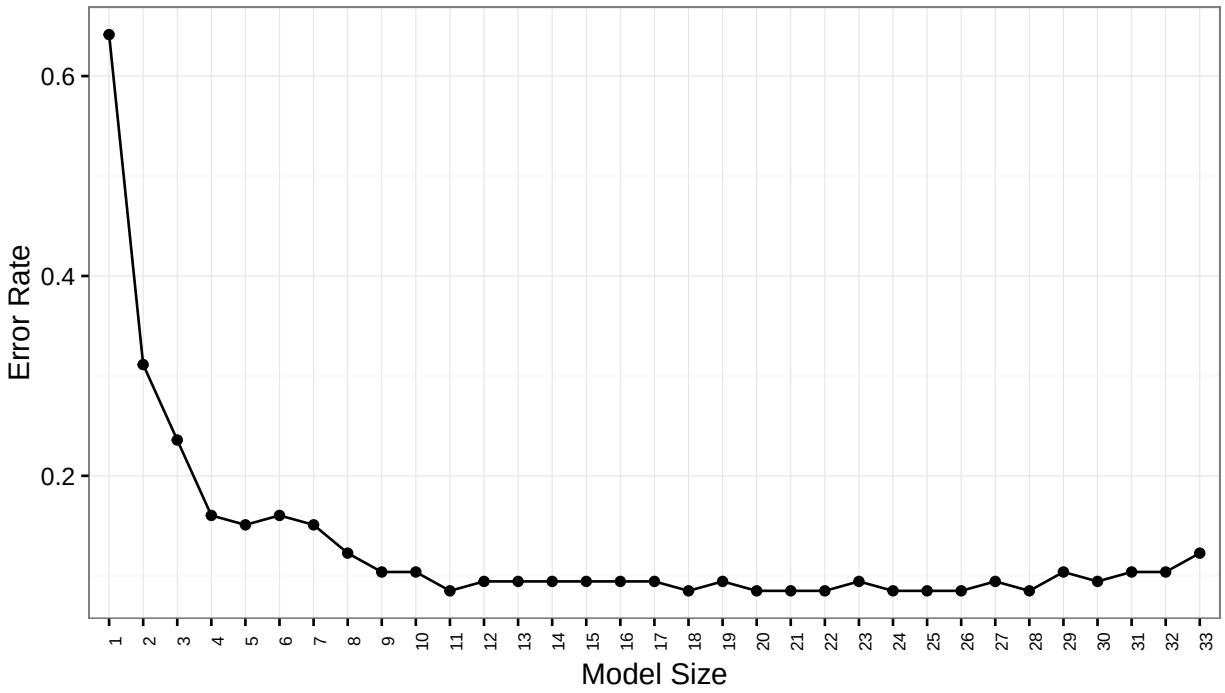

Figure 5.20: Best Model Size - Ionosphere Dataset

Looking at Figure 21 it seems that after eleven variables in the model, there isn't much change in the error.

	Model	Mean	Median	Var
1	RSSL	11.42	11.79	7.24
2	Adap-RSSL	11.06	10.38	6.83
3	RASSEL	11.06	10.38	6.83
4	ERASSEL	10.83	10.85	6.29

Table 5.8: Error Rate Summary by Model - Ionosphere Data

With the ionosphere dataset, ERASSEL performed better than the other three methods when looking at the average error rate, but it also had the worst median. All the variation of RSSL performed better than just plain RSSL. The results obtained for the ionosphere dataset of can be found can found on table 5.8 and Figure 22. Again, just as in regression, all four algorithms have very similar performance and shows that ERASSEL is performing just as well as the other algorithms. Now, let's have a look at the situation where Random Subspace learning comes in handy when we have more observations than variables ($p \ggg n$). The prostate dataset has genome information on patients that have prostate cancer, let's have a look at how those for algorithm perform on this dataset. When looking at the subspace size for this dataset, there wasn't much of a difference in the error from one size to the next; the error rate was oscillating between 30% and 32%. So for this problem we will use Breiman's recommendation of $d = \lceil \sqrt{p} \rceil$. So for this dataset, we have 501 variables, which means that the size of the subspace was set to 23. Let's now have a look at the results.

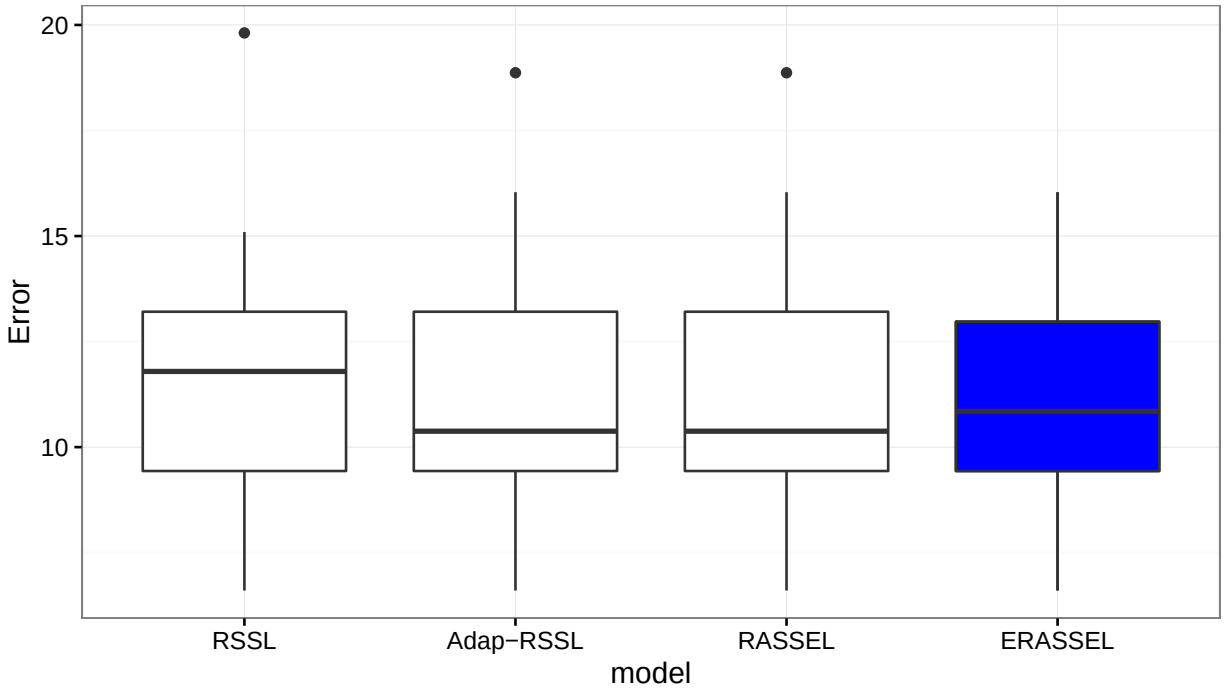

Figure 5.21: Boxplot of Errors by Model - Ionosphere Dataset

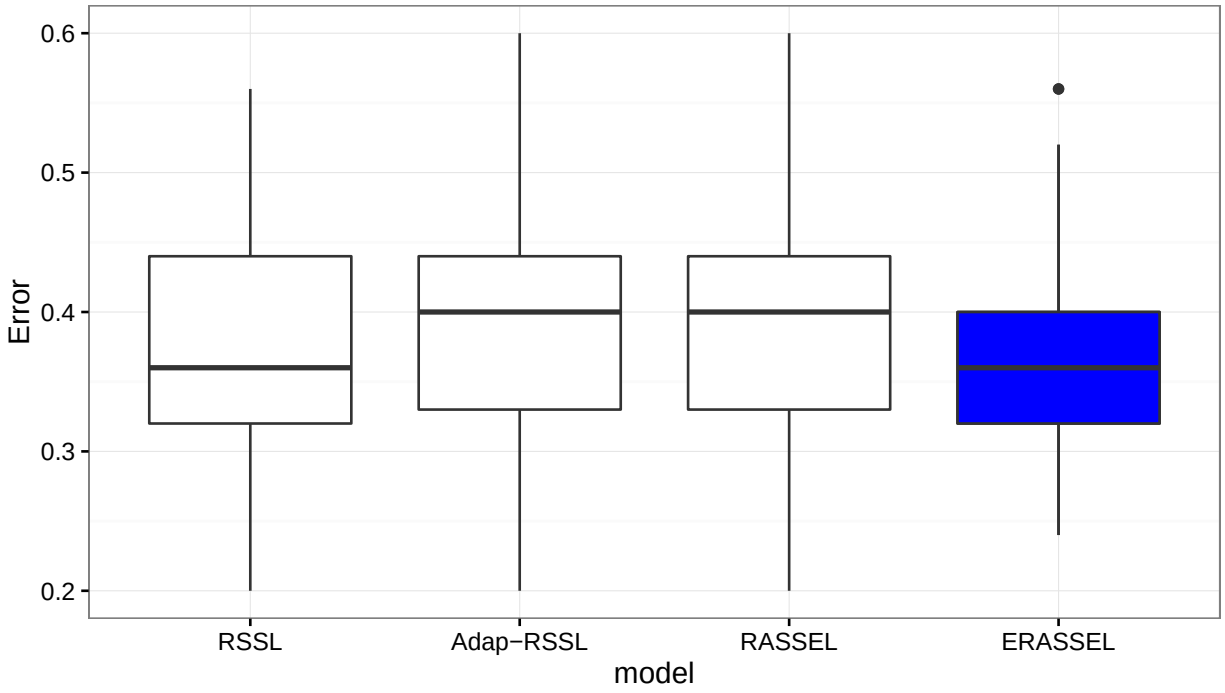

Figure 5.22: Boxplot of Errors by Model - Prostate Dataset

	Model	Mean	Median	Var
1	RSSL	0.38	0.36	0.01
2	Adap-RSSL	0.39	0.40	0.01
3	RASSEL	0.39	0.40	0.01
4	ERASSEL	0.36	0.36	0.01

Table 5.9: Error Rate Summary by Model - Prostate Data

As we can see, ERASSEL seems to be doing a good job on this dataset; it has the smaller median and smaller variation than all other algorithms, plain RSSL also seems to be doing well. Adap-RSSL and RASSEL are doing performed extremely similar. This dataset concludes our comparative results for this thesis. As we have seen from this results, it seems that RASSEL, using correlation between the response and the predictors, is a very good option when performing random subspace learning with regression. As for classification, all methods, including ERASSEL seem to be performing similarly well.

Chapter 6

Conclusion and Future Work

In this thesis, we have presented a new algorithm for ensemble learning, which is a common technique in Machine Learning, specially for when the task is mainly to make predictions. The focus of this new algorithm was Random Subspace learning, which focuses on selecting different subspaces of the feature space, and thus creating variation in the base learners. Random subspace learning is an important method for ensemble learning, as it allows for fitting model on conditions that might not be ideally for fitting common statistical models, namely when $n \ggg p$. This algorithm focuses assigning weights to variables in the feature space. The interesting aspect of different weighting schemes for random subspace learning is that important not to waste computing time by looking at variables that are not useful to predict our response.

The way this algorithm works is that as the ensemble grows, it update the variables weights based previous models of the ensemble to select the next subspace, that way giving more weight to variables that prove to be important in predicting the response and avoiding using variables that do not help on the prediction. This idea came from the much known evolution of species in the natural world, where good features survive throughout the environment and bad traits tend to die off, or least be less common. with that in mind, the algorithm was called Evolutionary Adaptive Random Subspace Learning.

We have shown that ERASSEL can be used for regression and classification tasks, the only adjustment needed is on how the updating of weight works, which have been presented in this work. The base learners used in order to perform the empirical test was GLM, which is widely applied in the statistical community and can be easily adapted to different situations, such as regression and classification. It is important to keep in mind that ERASSEL could be adapted to many other learners, such as classification and regression trees or SVM.

First by using simulated data, we were able to see how the algorithm works. We were able to see that as the number in the ensemble grows the algorithm is able to distinguish good predictor from bad predictors, mainly with orthogonal data. The other interesting thing on this simulation, it showed us that the weights of ERASSEL seem to mimic RASSEL static weights, when the ensemble is large enough. As shown in the ensemble learning chapter, the more variability we have in the ensemble the better, so this convergence might be useful in order to create a variability that RASSEL might not provide.

After looking how behind the curtains, we used simulated and real data for both regression and classification to compare it with plain RSSL, Adaptive RSSL, and RASSEL, which showed us that ERASSEl is comparable to algorithms already well established in the data mining community. Although, it was shown that RASSEL performs better than ERASSEL and the others for regression tasks. As for classification, ERASSEL was more comparable to RASSEL, and it perform very well on the dataset where we had more variables than observations.

As future, we would like to investigate different ways to update the variable weights as to be able to discern more between good and bar variables. We also want to insert a mechanism that can deal with highly correlated predictors, because as we saw - with simulated data - ERASSEL isn't able to perform well in this scenario

Appendices

Appendix A

Selected R Code

```
##############################################################################
#                                PACKAGES                                    #
##############################################################################
library(mlbench)
library(kernlab)
library(datasets)
library(sampling)
library(MASS)
library(ggplot2)
library(reshape2)
library(dplyr)


##############################################################################
#                                FUNCTIONS                                   #
#                       For Random Subspace Learning                         #
##############################################################################

##############################################################################
################################ Regression ##################################
##############################################################################



##### RSSL #########
rssl.lm<-function(response,data,d,L){
    n<-nrow(data)
    p<-ncol(data)
    resp.pos<-which(names(data)==response)
    mod.list<-NULL
```

```
    for( i in 1:L){
      sel.sub<-sample(c(1:p)[-resp.pos],d,replace = F)
      subs.data<-paste(names(data)[sel.sub],collapse = "+")
      form<-as.formula(paste0(response,"~",subs.data))
      m<-lm(form,data)
      mod.list<-c(mod.list,list(m))
    }
    return(mod.list)
}




#### Adaptive RSSL  uniform weight ####

rsslbag.lm<-function(response,data,d,L,size){
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  mod.list<-NULL

  for( i in 1:L){
    sel.obs<-sample(1:n,size,replace=T)
    sel.sub<-sample(c(1:p)[-resp.pos],d,replace = F)
    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))
    m<-lm(form,data[sel.obs,])
    mod.list<-c(mod.list,list(m))
  }

  return(mod.list)
}

#### Adaptive RASSEL ####

rassel.lm<-function(response,data,d,L,size){
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  cor.resp<-cor(data)[,response][-resp.pos]
  weight.sel<-cor.resp^2/sum(cor.resp^2)
  mod.list<-NULL

  for( i in 1:L){
    sel.obs<-sample(1:n,size,replace=T)
```

```
    sel.sub<-sample(c(1:p)[-resp.pos],d,replace = F,prob=weight.sel)
    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))
    m<-lm(form,data[sel.obs,])
    mod.list<-c(mod.list,list(m))
  }

  return(mod.list)
}




####### Evolutionary RASSEL ########

erassel.lm<-function(response,data,d,L,size){
  #### Step 1 : Making a pool of models to choose from
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  predictors<-c(1:p)[-resp.pos]
  burn.list<-NULL
  mod.list<-NULL
  weight.df<-data.frame(ids=1:ncol(data),V1=0,V2=0)
  weight.path<-NULL
  for( i in c(3:(L+2))){
    #weight.df[,i]<-rowMeans(weight.df[,-1],na.rm = T)
    preds.w <- rowMeans(weight.df[,-1],na.rm = T)
    weight.df[,i]<-preds.w
    preds.w[-resp.pos]<-exp(-preds.w[-resp.pos])/sum(exp(-preds.w[-resp.pos]))

    sel.obs<-sample(1:n,size,replace=T)
    sel.sub<-sample(predictors,d,replace = F,prob=preds.w[-resp.pos])

    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))

    m<-lm(form,data[sel.obs,])
    pred.l<-predict(m,data[-sel.obs,])

    acc<- mean((pred.l-data[-sel.obs,response])^2)  ## out of bag error
    mod.list<-c(mod.list,list(m))
    weight.df[,i]<-weight.df[,i-1]
    weight.df[sel.sub,i]<-acc
```

```
    weight.path<-cbind(weight.path,preds.w)
  }

  return(list(models=mod.list,weights=preds.w[-resp.pos],
  df=weight.df[-resp.pos,],w.path=weight.path[-resp.pos,]))
}


################################################################################
############################ Classification ####################################
################################################################################


##### RSSL ####
names(data)
rssl.glm<-function(response,data,d,L){
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  mod.list<-NULL
  for( i in 1:L){
    sel.sub<-sample(c(1:p)[-resp.pos],d,replace = F)
    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))
    m<-glm(form,data,family = binomial())
    mod.list<-c(mod.list,list(m))
  }
  return(mod.list)
}


#### Adaptive RSSL  uniform weight ####
rsslbag.glm<-function(response,data,d,L,size){
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  mod.list<-NULL

  for( i in 1:L){
    sel.obs<-sample(1:n,size,replace=T)
    sel.sub<-sample(c(1:p)[-resp.pos],d,replace = F)
    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))
    m<-glm(form,data[sel.obs,],family = binomial())
    mod.list<-c(mod.list,list(m))
```

```
  }
  return(mod.list)
}


#### Adaptive RASSEL ####
vaov<-function(resp,x,data){
  forml<-as.formula(paste0(resp,"~",x))
  a1<-aov(forml,data)
  a1<-summary(a1)
  return(a1[[1]]$`F value`[1])
}

rassel.glm<-function(response,data,d,L,size){
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  mod.list<-NULL
  f<-unlist(lapply(data[,-resp.pos],
                   function(v)summary(aov(Class~v,data))[[1]]$`F value`[1]))
  weight.sel<-f/sum(f)
  for( i in 1:L){
    sel.obs<-sample(1:n,size,replace=T)
    sel.sub<-sample(c(1:p)[-resp.pos],d,replace = F,prob=weight.sel)
    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))
    m<-glm(form,data[sel.obs,],family = binomial())
    mod.list<-c(mod.list,list(m))
  }
  return(mod.list)
}


erassel.glm2<-function(response,data,d,L,size){
  n<-nrow(data)
  p<-ncol(data)
  resp.pos<-which(names(data)==response)
  predictors<-c(1:p)[-resp.pos]
  burn.list<-NULL
  mod.list<-NULL
  weight.df<-data.frame(ids=1:ncol(data),V1=0,V2=0)

  for( i in c(3:(L+2))){
    weight.df[,i]<-rowMeans(weight.df[,-1],na.rm = T)
    preds.w <- exp(-weight.df[,i])/sum(exp(-weight.df[,i]),na.rm=T)
```

```
    sel.obs<-sample(1:n,size,replace=T)
    sel.sub<-sample(predictors,d,replace = F,prob=preds.w[-resp.pos])

    subs.data<-paste(names(data)[sel.sub],collapse = "+")
    form<-as.formula(paste0(response,"~",subs.data))

    m<-glm(form,data[sel.obs,],family = binomial())
    pred.g<-predict(m,data[-sel.obs,],type='response')
    pred.g<-ifelse(pred.g>0.5,1,0)

    ct<-table(pred.g,data[-sel.obs,response])## out of bag error
    acc<-sum(diag(prop.table(ct)))
    mod.list<-c(mod.list,list(m))

    weight.df[sel.sub,i]<-1-acc
  }
  return(list(models=mod.list,weights=preds.w[-resp.pos],
  df=weight.df[-resp.pos,]))
}


#############################################################################
#                                    DATA                                   #
#############################################################################

#############################################################################
############################### Real World Data ##############################
#############################################################################

data(spam)
data("Pima.tr")
data("Pima.te")
data("Ionosphere")

#############################################################################
########################## Simulated Data For Regression #####################
#############################################################################

mu<-round(runif(10, 0,5),2) #10 variables with means in [0,5]

###### non-correlated variables #####
sigma<-round(runif(10,1,3),2)
sigma.m<-`diag<-`(matrix(0, 10, 10), 1)*sigma
#### Generating data
```

```
X<-mvrnorm(300,mu,sigma.m)
# True function
eps<-rnorm(300,0,4)
y<-X[,1]+X[,4]+X[,5]+eps
sim.dat<-data.frame(cbind(y,X))


###### High Correlation between two variables  #####
d<-replicate(20,rnorm(20))
d<-cor(d)
sigma.m<-d

#### Generating data
X<-mvrnorm(300,mu,sigma.m)
# True function
eps<-rnorm(300,0,4)
y<-X[,1]+X[,4]+X[,5]+eps
sim.dat<-data.frame(cbind(y,X))


mu<-round(runif(10, 0,5),2) #10 variables with means in [0,5]
###### correlated data #####
d<-replicate(10,rnorm(10))
d[,7]<-d[,1]*2+rnorm(10) # X7 is now a linear function of X1
sigma.m<-cor(d)
#### Generating data
X<-mvrnorm(300,mu,sigma.m)
# True function
eps<-rnorm(300,0,4)
y<-X[,1]+X[,4]+X[,5]+eps
sim.dat<-data.frame(cbind(y,X))

##############################################################################
########################### Simulated Data For Logistic  #####################
##############################################################################

mu<-round(runif(10, 0,1),2) #10 variables with means in [0,5]
###### non-correlated data #####
sigma<-round(runif(10,1,3),2)
sigma.m<-`diag<-`(matrix(0, 10, 10), 1)*sigma
#### Generating data
X<-mvrnorm(300,mu,sigma.m)
# True function
eps<-rnorm(300,0,4)
y<-X[,1]+X[,4]+X[,5]
```

```
pii<-1/(1+exp(-y))

y.c<-rbinom(300,1,pii)
table(y.c)
sim.dat.c<-data.frame(cbind(y.c,X))
names(sim.dat.c)[1]<-"Class"




#############################################################################
#                                Example of  Replication                    #
#############################################################################

### This is an example of how to generate the comparative boxplots
### It will only be shown here for the simulated data for logistic regression
### The idea is the same for every dataset, which just needs to be changed
### inside the for() loop instead of `sim.data.c`.


e0<-e.rssl<-e.ada<-e.rassel<-e.erassel<-NULL # creating vectors to store errors
nsim<-50 # Number of replications
k<-7 # Size of base learners
for ( i in 1:nsim){
#statified sample
  st <- strata(sim.dat.c,stratanames=c("Class"),size=c(109,100),
            method="srswor")

  train.dt <- sim.dat.c[st$ID_unit,]
  test.dt <- sim.dat.c[-st$ID_unit,]

  glm1<-glm(Class~.,train.dt,family = binomial())
  preds1<-predict(glm1,test.dt,type="response")
  preds1<-ifelse(preds1>0.5,1,0)
  ct1<-prop.table(table(preds1,test.dt$Class))
  e0[i]<-1-sum(diag(ct1))

  rssl1<-rssl.glm("Class",data = train.dt,d =k, L=200)
  preds2<-lapply(rssl1,predict,test.dt,type="response")
  preds2<-do.call(cbind,preds2)
  preds2 <- rowMeans(preds2)
  preds2 <- ifelse(preds2>0.5,1,0)
  ct2<-prop.table(table(preds2,test.dt$Class))
  e.rssl[i]<-1-sum(diag(ct2))
```

```
  rssl2<-rsslbag.glm("Class",data = train.dt,d =k, L=200,size = 0.7*(nrow(train.dt)))
  preds3<-lapply(rssl2,predict,test.dt,type="response")
  preds3<-do.call(cbind,preds3)
  preds3 <- rowMeans(preds3)
  preds3 <- ifelse(preds3>0.5,1,0)
  ct3<-prop.table(table(preds3,test.dt$Class))
  e.ada[i]<-1-sum(diag(ct3))


  rssl3<-rassel.glm("Class",data = train.dt,d =k, L=200,size = 0.7*(nrow(train.dt)))
  preds4<-lapply(rssl3,predict,test.dt,type="response")
  preds4<-do.call(cbind,preds4)
  preds4 <- rowMeans(preds4)
  preds4 <- ifelse(preds4>0.5,1,0)
  ct4<-prop.table(table(preds4,test.dt$Class))
  e.rassel[i] <- 1-sum(diag(ct3))


  rssl4<-erassel.glm2("Class",data = train.dt,d=k, L=200,size = 0.7*(nrow(train.dt)))
  preds5<-lapply(rssl4$models,predict,test.dt,type="response")
  preds5<-do.call(cbind,preds5)
  preds5 <- rowMeans(preds5)
  preds5 <- ifelse(preds5>0.5,1,0)
  ct4<-prop.table(table(preds5,test.dt$Class))
  e.erassel[i] <- 1-sum(diag(ct4)) #0.0839971

  cat(i,":",e0[i],";",e.rssl[i],";",e.ada[i],";",e.rassel[i],";",e.erassel[i],"\n")
}

## Creating data frame with errors
class.error<-data.frame(
model=rep(c("Glm","RSSL","Adap-RSSL","RASSEL","ERASSEL"),each=nsim),
                        error=c(e0,e.rssl,e.ada,e.rassel,e.erassel))
# Reording factor levels
class.error$model <- factor(class.error$model,
levels=c("Glm","RSSL","Adap-RSSL","RASSEL","ERASSEL"))

# Creates table with summary measures for each type of modeling

class.error %>%
  group_by(Model=model) %>%
  summarise(Mean=mean(error),Median=median(error),Var=var(error)
```

Chapter 7

References

Akaike, H. 1974. "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* 19 (6): 716–23.

Baba, Nura Muhammad, Mokhairi Makhtar, Syed Abdullah Fadzli, and Mohd Khalid Awang. 2015. "Current Issues in Ensemble Methods and Its Applications." *Journal of Theoretical and Applied Information Technology* 81 (2).

Breiman, Leo. 1984. *Classification and Regression Trees*. Belmont, Calif: Wadsworth International Group.

———. 1996a. "Bagging Predictors." *Machine Learning* 24 (2). Springer: 123–40.

———. 1996b. "Stacked Regressions." *Machine Learning* 24 (1). Springer: 49–64.

———. 2001. "Random Forests." *Machine Learning* 45 (1). Springer: 5–32.

Brown, Gavin, Jeremy Wyatt, Rachel Harris, and Xin Yao. 2005. "Diversity Creation Methods: A Survey and Categorisation." *Information Fusion* 6 (1): 5–20.

Bryll, Robert, Ricardo Gutierrez-Osuna, and Francis Quek. 2003. "Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets." *Pattern Recognition* 36 (6): 1291–1302. doi:http://dx.doi.org/10.1016/S0031-3203(02)00121-8.

Clarke, Bertrand, Ernest Fokoué, and Hao Helen Zhang. 2009. *Principles and Theory for Data Mining and Machine Learning*. Springer Science & Business Media.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support Vector Machine." *Machine Learning* 20 (3): 273–97.

Coussement, Kristof, and Koen W. De Bock. 2013. "Customer Churn Prediction in the Online Gambling Industry: The Beneficial Effect of Ensemble Learning." *Journal of Business Research* 66 (9): 1629–36.

Cover, Thomas M, and Peter E Hart. 1967. "Nearest Neighbor Pattern Classification." *Information Theory, IEEE Transactions on* 13 (1). IEEE: 21–27.

Dahl, David B. 2015. *Xtable: Export Tables to LaTeX or HTML*. https://CRAN.R-project.org/package=xtable.

Darwin, Charles. 1929. *The Origin of Species*. John Murry London.

Dasarathy, B. V., and B. V. Sheela. 1979. "A Composite Classifier System Design: Concepts and Methodology." *Proceedings of the IEEE* 67 (5): 708–13.

Dobson, Annette J., and Adrian G. Barnett. 2008. *An Introduction to Generalized Linear Models*. 3rd ed. Boca Raton: CRC Press.

Faraway, Julian J. 2006. *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Boca Raton: Chapman & Hall/CRC.

Fokoué, Ernest. 2016. "Prediction Error Reduction Function as a Variable Importance Score." *British Journal of Mathematics & Computer Science* 15 (3). doi:10.9734/BJMCS/2016/23945.

Fokoué, Ernest, and Mohamed Elshrif. 2015. *Improvement of Predictive Performance via Random Subspace Learning with Data Driven Weighting Schemes*. Rochester Institute of Techonology.

Fokoué, Ernest, and VI Ly. 2014. *Frequent Approximation of the Bayesian Posterior Inclusion Probability by Stochastic Subsampling*. Rochester Institute of Technology. http://scholarworks.rit.edu/article/1751.

Freund, Y. 1995. "Boosting a Weak Learning Algorithm by Majority." *Information and Computation* 121 (2): 256–85. doi:http://dx.doi.org/10.1006/inco.1995.1136.

Freund, Yoav, and Robert E. Schapire. 1997. "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting." *Journal of Computer and System Sciences* 55 (1): 119–39.

Gutierrez-Osuna, Ricardo. "Ensemble Learning." Lecture. Texas A&M. http://research.cs.tamu.edu/prism/lectures/pr/pr_l25.pdf.

Hastie, Trevor, Robert Tibshirani, and J. H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.

Ho, Tin Kam. 1998. "The Random Subspace Method for Constructing Decision Forests." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20 (8): 832–44.

Karatzoglou, Alexandros, Alex Smola, Kurt Hornik, and Achim Zeileis. 2004. "Kernlab – an S4 Package for Kernel Methods in R." *Journal of Statistical Software* 11 (9): 1–20. http://www.jstatsoft.org/v11/i09/.

Kittler, J., M. Hatef, R. P. W. Duin, and J. Matas. 1998. "On Combining Classifiers." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (3): 226–39.

Kramer, Oliver. 2013. *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Springer.

Krogh, Anders, Jesper Vedelsby, and others. 1995. "Neural Network Ensembles, Cross Validation, and Active Learning." *Advances in Neural Information Processing Systems* 7. MORGAN KAUFMANN PUBLISHERS: 231–38.

Kuncheva, Ludmila I. 2014. *Combining Pattern Classifiers: Methods and Algorithms*. 2nd ed. Hoboken, New Jersey: Wiley.

Lai, Carmen, Marcel J.T. Reinders, and Lodewyk Wessels. 2006. "Random Subspace Method for Multivariate Feature Selection." *Pattern Recognition Letters* 27 (10): 1067–76. doi:http://dx.doi.org/10.1016/j.patrec.2005.12.018.

Leisch, Friedrich, and Evgenia Dimitriadou. 2010. *Mlbench: Machine Learning Benchmark Problems*. https://CRAN.R-project.org/package=mlbench.

McCullagh, Peter, and John A Nelder. 1989. *Generalized Linear Models*. Vol. 37. CRC press.

Neter, John, William Wasserman, and Michael H. Kutner. 1989. *Applied Linear Regression Models*. 2nd ed. Homewood, Ill: Irwin.

Peter J. Bickel, Alexandre B. Tsybakov, Ya'acov Ritov. 2009. "Simultaneous Analysis of Lasso and Dantzig Selector." *The Annals of Statistics* 37 (4). Institute of Mathematical Statistics: 1705–32. http://www.jstor.org/stable/30243685.

Plackett, R. L. 1950. "Some Theorems in Least Squares." *Biometrika* 37 (1/2): 149.

R Core Team. 2015. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Schwarz, Gideon et al. 1978. "Estimating the Dimension of a Model." *The Annals of Statistics* 6 (2). Institute of Mathematical Statistics: 461–64.

Stamey, Thomas A, John N Kabalin, John E McNeal, Iain M Johnstone, Fuad Freiha, Elise A Redwine, and Norman Yang. 1989. "Prostate Specific Antigen in the Diagnosis and Treatment of Adenocarcinoma of the Prostate. II. Radical Prostatectomy Treated Patients." *The Journal of Urology* 141 (5): 1076–83.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 267–88.

Tillé, Yves, and Alina Matei. 2015. *Sampling: Survey Sampling*. https://CRAN.R-project.org/package=sampling.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Fourth. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Wezel, Michiel van, and Rob Potharst. 2007. "Improved Customer Choice Predictions Using Ensemble Methods." *European Journal of Operational Research* 181 (1): 436–52.

Wickham, Hadley. 2007. "Reshaping Data with the reshape Package." *Journal of Statistical Software* 21 (12): 1–20. http://www.jstatsoft.org/v21/i12/.

———. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://had.co.nz/ggplot2/book.

Wickham, Hadley, and Romain Francois. 2015. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Xie, Yihui. 2013. "Knitr: A General-Purpose Package for Dynamic Report Generation in R." *R Package Version* 1 (7): 1.

———. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." *Implement Reprod Res* 1: 20.

———. 2015. *Dynamic Documents with R and Knitr*. Vol. 29. CRC Press.

Zhang, Cha, and Yunqian Ma. 2012;2015. *Ensemble Machine Learning: Methods and Applications*. 1. Aufl. New York: Springer.