A stacked generalization approach to parameter selection for supervised-learning problems

Candidate Number: 43122

Dissertation submitted to the
Department of Management
for the degree of
Master of Science

1 September 2016

Abstract

This paper is divided into two main sections. The first section formally introduces stacked generalization through a review of the key literature on that subject. Favourable conditions for the use of stacked generalization are discussed, as well as known issues and variations of the original technique introduced by Wolpert (1992).

In the second section, we try to use stacked generalization for a problem where it is not traditionally employed: parameter selection in a supervised-learning context. Through experimentation, we find that when a parameter selection technique such as cross-validation is unstable, it is possible to "stack" the fitted values generated with different parameters and obtain lower errors than when simply choosing the best parameter. For our dataset, we also find that of all the studied learning schemes, the lowest mean error rate is obtained with a regression tree as the meta-learner, while a ridge regression returns the most interpretable final fitted values.

Contents

1 Introduction

This paper is divided into two main sections. The first section is concerned with doing a literature review of stacked generalization. In a few words, in a supervised-learning context, stacked generalization is a technique used to learn from the fitted values of different learning schemes. It does so by using the fitted values as inputs in another learning scheme. The first section will formally introduce stacked generalization to the reader through a review of the most important literature on the subject.

The second section is concerned with formulating an hypothesis, performing an experiment to confirm or reject the hypothesis and, finally, analysing the results. Most of the knowledge in the field of stacked generalization is derived from experimentation (e.g. Wolpert (1992), LeBlanc and Tibshirani (1996), Ting and Witten (1999), etc.). Thus, this paper is in line with the current literature: acknowledge what is known, formulate a new hypothesis and test it through experimentations. For this paper, the hypothesis in question has to do with the suitability of using stacked generalization for a problem where it is not traditionally employed: parameter selection.

Indeed, a number of learning schemes require one to select a parameter before fitting the model. These include, for example, the bandwidth of a kernel regression, the depth of a decision tree or the number of layers of a neural network. The hypothesis for this experiment is that if the procedure (e.g. cross-validation) to select such parameter is unstable, then it can be beneficial to aggregate the fitted values of the learning scheme using different parameters rather than trying to pick the "best" parameter. This "aggregation" can be done by using the fitted values as inputs into another model. In other words, by *learning* at the meta level. To our knowledge, this hasn't been studied yet, or at least not in this specific context.

The second section will present the results of the experiment, confirm or reject the hypothesis and draw conclusions. Finally, we will conclude this paper by discussing opportunities for further research.

2 Literature review

The first section of this paper is concerned with doing a literature review of stacked generalization (often simply referred to as *stacking*). First, the idea of combining forecasts, and more specifically stacked generalization, will be formally introduced. Then, an accompanying notation will be presented and different ways to combine the models will be addressed. Second, we will evaluate in which circumstances stacking can be useful. Third, some known issues of stacking will be discussed. Finally, a selected number of variations of the original algorithm will be presented.

2.1 Combining models

The idea of combining models has a long history. As Armstrong (2001, p. 417) notes is his book *Principles of Forecasting*, "important papers on this approach date back at least as far as the mid-1950s (e.g. Cronbach and Meehl, 1955)".

Intuitively, combining the results of different models for regression or classification tasks makes sense. If one sees a fitted model as an "expert" and is presented with new data, then it seems logical to ask multiple "experts" before making a decision. But how one should decide which experts to consult? How should their predictions be aggregated? As Wolpert (1992, p. 243) notes, "one is always implicitly presented with the problem of how to address this multiplicity of possible generalizers".

Techniques such as cross-validation allow us to select the single best-performing model, but it is a winner-takes-all strategy and doesn't take advantage of the full span of predictions. Other ensemble techniques, such as bagging (Breiman, 1996a), average the outputs of the different experts. As we will see, stacked generalization is a technique which allows one to aggregate the output of different models by learning at the meta-level as well (instead of simply averaging the results, for example).

2.2 About stacked generalization

The concept of stacked generalization as we know it was first introduced by Wolpert (1992). Then, Breiman (1996b) built on this idea to create stacked regression. However, LeBlanc and Tibshirani (1996) note that an identical idea was proposed by Stone (1974) under the name "model-mix". Nevertheless, it appears that the name "stacked generalization", or simply "stacking", is commonly used within the machine learning community.

The idea of stacking consists of combining different learning schemes in order to obtain better results. As such, it falls into the category of *ensemble methods*, like boosting (Breiman et al., 1998; Schapire, 1990) and bagging (Breiman, 1996a). The term "better results" is purposefully vague here because stacking has been used successfully in regression tasks (e.g. Breiman (1996b)), classification tasks (e.g. Ting and Witten (1999)) and even for unsupervised learning problems (e.g. Smyth and Wolpert (1997)). In the present paper, we shall be concerned with regression and classification problems. Consequently, when there is a mention of "better results", what is referred to is most likely a lower prediction error (e.g. lower root mean squared error, or RMSE) or lower classification error rate.

In its simplest form, as presented in Wolpert (1992), stacking consists of a two-step approach. First, one or more learning schemes (or "generalizers" as Wolpert calls them) are

fitted to some data. This is the *level-0*. Then, the fitted values (predictions) of those level-0 generalizers are used as explanatory variables (inputs) in another generalizer. This is the *level-1*. The final predictions are the fitted values of the level-1 generalizer. Algorithm 1 shows the basic skeleton of stacked generalization.

Algorithm 1: Basic stacking algorithm

```
// level-0
foreach level-0 generalizer do
    Fit a model on the data
    Generate fitted values
end
// level-1
Fit the level-1 model using the fitted values of level-0 as inputs and the response
variable of level-0 as response variable
Generate the final fitted values
```

In the next subsections, stacked generalization will be formally explained and an accompanying notation will be presented. Finally, different learning methods at level-1 will be explored.

2.2.1 Explanation and notation

In the notation below, the superscript (0) is used to designate level-0, and (1) to designate level-1.

- Let D be a dataset comprised of a response variable (class) $Y^{(0)}$ and some explanatory variables $X^{(0)}$. Example of such a dataset:

ID	$X_1^{(0)}$	$X_2^{(0)}$	$Y^{(0)}$
1	10.6	6.2	5.0
2	3.2	7.8	4.2
3	5.9	2.4	3.7

- Let I be an ensemble of learning methods. Say, for example, a linear regression (LR), a random forest (RF1), another random forest (RF2) using different parameters, and a ridge regression (RR).
- Let $G^{(0)}$ be the level-0 generalizers (fitted models) and $G_i^{(0)} \in G^{(0)}$ be the generalizer using learning method $i \in I$ fitted on D. Let $\hat{Y}_i^{(0)}$ be the fitted values (predictions) of $G_i^{(0)}$.
- Let $G^{(1)}$ be the level-1 generalizer, $\hat{Y}^{(1)}$ be the fitted values of $G^{(1)}$ and $X^{(1)}$ be the inputs of $G^{(1)}$.

As explained earlier, simply put, the idea of stacking is that the $\hat{Y}_i^{(0)}$ become $X^{(1)}$. In other words, the fitted values of $G^{(0)}$ become the inputs of $G^{(1)}$. Furthermore, $Y^{(0)} = Y^{(1)}$, which makes sense because we are trying to predict the same thing on both levels. The final predictions of this whole process are $\hat{Y}^{(1)}$. Let's revisit Algorithm 1, this time using the above notation.

Algorithm 2: Basic stacking algorithm using notation

```
// level-0
foreach i ∈ I do
    Fit G_i^(0) on D
    Generate Ŷ_i^(0)
end
// level-1
Fit G^(1) using all Ŷ_i^(0) as inputs and Y^(1) as response variable
Generate Ŷ^(1)
```

Graphically, the process looks like Figure 1:

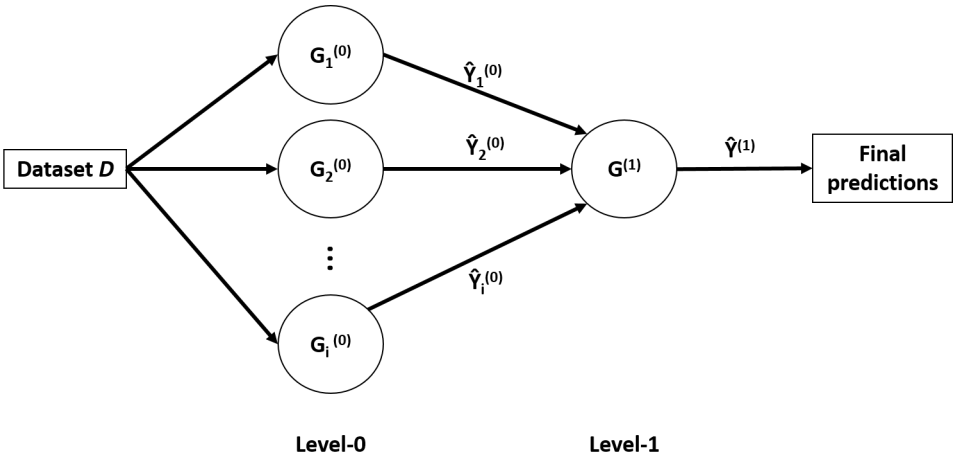


Figure 1: Schema of the stacked generalization process

Table 1 gives a (fictional) example of the stacking procedure. Notice that the $\hat{Y}_i^{(0)}$ are equal to the $X_i^{(1)}$.

	ID	$X_1^{(0)}$	$X_2^{(0)}$	$Y^{(0)}$	$\hat{Y}_{LR}^{(0)}$	$\hat{Y}_{RF1}^{(0)}$	$\hat{Y}_{RF2}^{(0)}$	$\hat{Y}_{RR}^{(0)}$
Level-0	1	10.6	6.2	5.0	4.9	5.1	5.0	4.8
	2	3.2	7.8	4.2	4.6	4.2	4.3	4.3
	3	5.9	2.4	3.7	3.7	3.5	3.8	3.9

	ID	$X_1^{(1)}$	$X_2^{(1)}$	$X_3^{(1)}$	$X_4^{(1)}$	$Y^{(1)}$	$\hat{Y}^{(1)}$
Level-1	1	4.9	5.1	5.0	4.8	5.0	5.0
	2	4.6	4.2	4.3	4.3	4.2	4.2
	3	3.7	3.5	3.8	3.9	3.7	3.7

Table 1: Stacking example

In practice, the fitted values can be generated with K-fold cross-validation so that the instances for which fitted values are generated are out-of-sample. This requires some adjustments to the notation and to Algorithm 2:

- Let $D_{M\times(N+1)}$ be a dataset containing M instances, N explanatory variables and 1 response variable. In the example of Table 1, $M = 3$ and $N = 2$.

- Let K be the number of folds (integer between 2 and M).

- Let $D^{[k]}$ be the subset of D containing the observations assigned to fold $k \in \{1, 2, \ldots, K\}$ and $D^{[-k]}$ be the subset of D containing the observations *not* assigned to k.

- Let δ be a dataset created at level-1 which contains M instances, I explanatory variables (all $\hat{Y}_i^{(0)}$) and one response variable (Y). $\delta^{[k]}$ is the subset of δ containing the observations assigned to fold $k \in \{1, 2, \ldots, K\}$ and $\delta^{[-k]}$ is the subset of δ containing the observations *not* assigned to k.

Then...

Algorithm 3: Stacking using K-fold cross-validation

```
// level-0
foreach i in I do
    Randomly assign each of the M instances of D to a fold k ∈ {1, 2, ..., K} with
    probability 1/K
    foreach k ∈ {1, 2, ..., K} do
        Fit G_i^(0) on D^[-k]
        Use G_i^(0) to generate fitted value for D^[k]
    end
    Store fitted values in Ŷ_i^(0)
end
// level-1
Create new dataset δ using all Ŷ_i^(0) and Y
Randomly assign each of the M instances of δ to a fold k ∈ {1, 2, ..., K} with
probability 1/K
foreach k ∈ {1, 2, ..., K} do
    Fit G^(1) on δ^[-k]
    Use G^(1) to generate fitted value for δ^[k]
    Store fitted values in Ŷ^(1)
end
Return Ŷ^(1)
```

Note that Wolpert (1992) suggests to split the original dataset D in $I + 1$ parts: one per $G_i^{(0)}$ and one for $G^{(1)}$. This is obviously doable, but it implies that D has a fairly large number of instances.

Because it uses cross-validation, Algorithm 3 has the advantage of being suitable for a smaller M. The process is fairly general and allows one to select any $G^{(1)}$. In the next subsections, different models and techniques for the level-1 generalizer $G^{(1)}$ will be explored.

2.2.2 Linear stacking

One way to combine the level-0 generalizers is linearly. Indeed, Breiman (1996b) studied this method and came up with a few conclusions which we will address here.

First, in a relatively basic form, linear combination can be achieved by using a linear regression as $G^{(1)}$. For an instance $m \in M$, the resulting final fitted value $\hat{Y}_m^{(1)}$ would thus be defined as

$$\hat{Y}_m^{(1)} = \alpha + \sum_{i=1}^{I} \beta_i \times G_i^{(0)}(X_m^{(0)})$$

or similarly (1)

$$\hat{Y}_m^{(1)} = \alpha + \sum_{i=1}^{I} \beta_i \times \hat{Y}_{i,m}^{(0)}$$

where α is the intercept and β_i are the coefficients of the linear regression $G^{(1)}$.

Second, some authors suggests ways to "regularize" the coefficients in order to achieve better and/or more interpretable results. Notably, LeBlanc and Tibshirani (1996) mention three regularization techniques: non-negativity, shrinkage and sum-to-one. The authors affirm that when experimenting with simulated data, "regularization substantially reduces the model error of the combination methods" (p. 1646).

Breiman (1996b), for example, first suggests to both restrain the domain of the coefficients to positive numbers (Lawson and Hanson (1995) developed an algorithm to solve non-negative coefficient least-squares problem) *and* to add a constraint so that coefficients sum to one. This, he argues, has the advantage of restraining the final predictions to the domain of the level-0 predictions. However, Breiman later shows that the sum-to-one constraint is largely unnecessary: he obtains very similar results with and without it.

While Breiman (1996b) seems to suggest that the non-negativity constraint is crucial to ensure that the stacking process will yield better results than simply selecting the best level-0 generalizer, Ting and Witten (1999) find (on classification problems) almost indistinguishable error rates when using non-negativity constraint and when *not* using it:

> "The non-negativity constraints found necessary by Breiman in regression are found to be irrelevant to improve predictive accuracy in our classification situation." (Ting and Witten, 1999, p.272)

That being said, the authors acknowledge that the non-negativity constraint has the advantage of making the coefficients more interpretable. Because they are all positive, one can more easily compare the relative importance of the level-0 generalizers in the level-1 final prediction.

In conclusion, it appears that coefficient regularization in linear stacking falls under what Wolpert (1992) calls "black art". In other words, authors do not seem to agree on whether or not regularization is necessary and/or desirable.

2.2.3 Non-linear stacking

The previous section dealt with a linear combination of the level-0 generalizers and its variants. Obviously, one can choose whatever learning scheme he/she sees fit as $G^{(1)}$, linear or not.

Ting and Witten (1999) compared the following level-1 generalizers: decision tree, K-Nearest Neighbours (K-NN), Naive Bayes and a linear model. The comparisons were made on 10 different datasets, 2 being artificial and 8 being real-world data. They found that when benchmarked against the best level-0 generalizer using cross-validation (BestCV), the linear model performs best. Indeed, the linear model performs better than BestCV on 9 out of 10 datasets and performs as well as BestCV on the tenth one. When the level-1 generalizers are compared against each-other, the linear model wins on 8 datasets, Naive Bayes on 1 and K-NN on 2 (there is a tie between K-NN and linear).

Although the results of Ting and Witten (1999) seem to show that sticking with linear models for $G^{(1)}$ generally works best, it is important to mention that other real-life applications have shown that non-linear methods can successfully be employed. For example, Ness et al. (2009) use support vector machines as the level-0 *and* level-1 generalizers to improve music tag annotation. Furthermore, Hu and Tsoukalas (2003) successfully use artificial neural networks as the level-0 *and* level-1 generalizers to get a better understanding of consumer choices. Note that we do not know if the two examples above would have worked better, had the authors used a linear model as the level-1 generalizer. Also note that obviously, one is not forced to use the same type of model for level-0 and level-1 when using a non-linear level-1 generalizer.

Once again, choosing the appropriate level-1 generalizer might fall under what Wolpert (1992) calls "black art", in the sense that we cannot know for sure before-hand which learning schemes will work best. However, experiments from Ting and Witten (1999) indicate that sticking with a linear approach is a safe bet.

2.3 When does stacking work?

By now we have seen a number of cases where stacked generalization was employed to make more accurate predictions or classifications. Thus, in this section, we shall not be concerned any more by *if* or *how* does stacking work, but *when* (in which circumstances).

2.3.1 Weakly correlated level-0 ouputs

Breiman notes that "the biggest gains [when using stacked generalization] came when dissimilar sets of predictors were stacked [...]. The more similar the predictors, the less advantage there is in stacking" (Breiman, 1996b, p.51). The intuition behind this is that there is a greater potential for improvement when combining models which generate dissimilar fitted values than when combining models which generate highly correlated fitted values.

Ting and Witten (1999) demonstrate this with an interesting experiment. First, they fit a number of level-0 generalizers on 10 different datasets (the same as in Section 2.2.3). For each dataset, they compute the number of standard errors (#SE) between the error rates of the worst performing and best performing (BestCV) level-0 generalizer. Consequently, a larger #SE means that the level-0 generalizers have dissimilar outputs. The table in Ting

Dataset	#SE	BestCV	Majority	MLR
Horse	0.5	17.1	**15.0**	15.2
Splice	2.5	4.5	4.0	**3.8**
Abalone	3.3	40.1	39.0	**38.3**
Led24	8.7	32.8	31.8	**31.3**
Credit	8.9	17.4	**16.1**	16.2
Nettalk(s)	10.8	12.7	12.2	**11.5**
Coding	12.7	25.0	23.1	**20.7**
Waveform	18.7	17.1	19.5	**16.8**
Euthyroid	26.3	**1.9**	8.1	**1.9**
Vowel	242.0	2.6	13.0	**2.5**

Table 2: Table 8 of Ting and Witten (1999) reproduced. Average error rates of BestCV, Majority Vote and MLR (multi-linear response), along with the number of standard error (#SE) between BestCV and the worst performing level-0 generalizer.

and Witten (1999, p. 281) showing the error rates and the #SE on the different datasets is reproduced in Table 2.

According to Breiman (1996b), we should expect the linear stacking (MLR) to work significantly better than Majority Vote when #SE is higher. Indeed, the authors find that "MLR performs significantly better in the five datasets that have large #SE values, but in only one of the other cases" (Ting and Witten, 1999, p. 281).

LeBlanc and Tibshirani (1996) seem to take a similar stand. When working on Boston housing data, they argue that they expect "a smaller improvement [...] because in this case there are a large number of observations and *not a large number of predictors, and several of the predictors are strongly associated with the response*" (p. 1646). Logically, they also state that they "would expect a larger improvement [...] if there were a larger number of *weakly predictive covariates* and/or a smaller training set size" (p. 1647).

2.3.2 Unstable model selection

Breiman (1996a) suggests that combining models would help when selection of the best model is unstable. For example, one traditional way to evaluate the performance of a model is through cross-validation. If small changes in the data make the performance of one(many) model(s) vary substantially and, as a result, the preferred model varies from fold to fold, it may be a sign that the problem at hand is best solved with an ensemble of models rather than with a single learning scheme.

One logical extension of this idea is parameter selection. Indeed, a number of learning schemes require one to select some parameter before executing the corresponding algorithm. Decision trees (number of leaves), kernel regressions (bandwidth) and neural networks (number of layers), for example, come to mind. A "traditional" approach would be to pick such a parameter based on cross-validation scores. This is a "winner-takes-all" approach. The argument could be made that if the parameter selection process is unstable, perhaps combining learning schemes using different parameters could yield better results than simply using the best parameter. This idea will be the subject of the second part of this paper.

2.4 Known issues

When combining the level-0 generalizers linearly, one problem that arises is that if those generalizers are any good, the inputs of level-1 ($X^{(1)}$) will be highly correlated. To address this problem, Breiman (1996b) suggests to use a ridge regression (Hoerl and Kennard, 1970) instead of the more traditional Ordinary Least Squares approach. When combined with a non-negative constraint on the coefficients, he argues that the resulting predictor "appears to almost always have lower prediction error than the single prediction $\hat{Y}_i^{(0)}$ having lowest cross-validation error" (Breiman, 1996b, p.50).

Other issues in stacked generalization have been mentioned earlier in this paper. Notably, Wolpert (1992) calls "black art" what has to do with (1) choosing the type of generalizer for level-1 and (2) choosing the types of attributes that should be used for $G^{(1)}$'s inputs.

Regarding the first point, we saw that Ting and Witten (1999) came to the conclusion that combining the level-0 linearly was the best approach, but we also saw that other methods have been used successfully. Furthermore, authors do not seem to agree on how to implement linear combination, Breiman (1996b) arguing that a non-negativity constraint is essential in order to guarantee that the stacking process yields better results than its best underlying model, while Ting and Witten (1999) found that the non-negativity constraint is unnecessary.

Regarding the second point, in the context of classification tasks, authors seem to agree that including probabilities fetched by the $G_i^{(0)}$, and not just the final classification, yields better results. Such findings were made by Ting and Witten (1999) and Džeroski and Ženko (2004), notably. Here again, many authors have suggested their own flavour of Wolpert (1992)'s original stacking technique (including Wolpert himself), which brings us to the next subsection.

2.5 Variations of stacking

As pointed out earlier, experts do not always agree when it comes to stacked generalization. Probably because the technique is quite general, many researchers and practitioners have come up with their own version of stacking. Usually these versions try to address one or more of the "issues" mentioned above. This subsection does *not* try to produce an exhaustive list of all the bells and whistles that have been added to stacking over the years, but rather tries to introduce the reader to a selected number of variations which have proven to be popular and/or useful amongst authors.

2.5.1 Predicting the error

Wolpert (1992) himself, in his paper which introduced stacked generalization, included a section called *Extensions and variations*. One approach that he suggested is that the level-1 generalizer could try to predict the *error* of one the level-0 generalizers, and not the target variable itself. Say that, using the example in Table 3 below, we want to use stacking to predict the error of the linear regression (LR). So, using the proposed notation:

$$Y^{(1)} = Y^{(0)} - \hat{Y}_{LR}^{(0)} \tag{2}$$

Level-0

ID	$X_1^{(0)}$	$X_2^{(0)}$	$Y^{(0)}$	$\hat{Y}_{LR}^{(0)}$	$\hat{Y}_{RF1}^{(0)}$	$\hat{Y}_{RF2}^{(0)}$	$\hat{Y}_{RR}^{(0)}$
1	10.6	6.2	5.0	4.9	5.1	5.0	4.8
2	3.2	7.8	4.2	4.6	4.2	4.3	4.3
3	5.9	2.4	3.7	3.7	3.5	3.8	3.9

Level-1

ID	$X_1^{(1)}$	$X_2^{(1)}$	$X_3^{(1)}$	$X_4^{(1)}$	$Y^{(1)}$	$\hat{Y}^{(1)}$
1	4.9	5.1	5.0	4.8	0.1	0.1
2	4.6	4.2	4.3	4.3	-0.4	-0.4
3	3.7	3.5	3.8	3.9	0.0	0.0

Table 3: Stacking example where the level-1 generalizer predicts the level-0 error of the linear regression (LR)

Given new data, the final prediction P of this technique would be

$$P = G_{LR}^{(0)}(X^{(0)}) + \alpha * G^{(1)}(X^{(1)}) \quad (3)$$

$$P = \hat{Y}_{LR}^{(0)} + \alpha * \hat{Y}^{(1)} \quad (4)$$

where $\alpha \in [0; 1]$ is a "parameter which determines to what extent we use stacked generalization and to what extent we simply use $G^{(0)}$ by itself" (Wolpert, 1992, p.30). In the context of the example of Table 3, for $\alpha = 1$:

$$P = \begin{bmatrix} 4.9 \\ 4.6 \\ 3.7 \end{bmatrix} + 1 * \begin{bmatrix} 0.1 \\ -0.4 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 5.0 \\ 4.2 \\ 3.7 \end{bmatrix}$$

2.5.2 More than one level

So far we only addressed stacked generalization using so-called level-0 and level-1. In practice, nothing retrains someone from adding one or more levels to the stacking structure. After all, the stacking process is a generalizer in itself, which can feed fitted values into another stacked generalization (and so on).

This idea was also proposed by Wolpert (1992). He mentioned that because such a structure would create a network, "usual network games (e.g. back-propagation) can then be applied" (Wolpert, 1992, p.31). However it remains to be seen if this approach can actually be useful. Wolpert himself only mentioned that this would be a possible extension of stacked generalization without digging deeper into it.

2.5.3 Blending

The term *blending* seems to be loosely used amongst practitioners as a synonym for combining generalizers in a similar (or exactly the same) way as stacked generalization. Here we briefly explain the concept as it was proposed in Töscher et al. (2009), authored by members of the winning team of the famous Netflix Grand Prize[1].

[1]See http://netflixprize.com for more details on the Netflix Grand Prize

The authors argue that, in a context where the accuracy of a generalizer is the only concern (like in the Netflix challenge), then the decision to add an extra $G_i^{(0)}$ to the "blender" should not depend on $G_i^{(0)}$'s performance per se, but on the resulting performance of $G^{(1)}$ once the extra $G_i^{(0)}$ is added.

In this context, given that there is a large number of potential $G_i^{(0)}$, trying every combination of $G_i^{(0)}$ quickly becomes impractical, computationally speaking. So the authors suggest to iteratively add $G_i^{(0)}$ to the blender and keep it there only if it improves the performance of $G^{(1)}$. This probably leads to a local solution, but a local solution that was nevertheless good enough to win the Netflix challenge. For more details, see *The BigChaos Solution to the Netflix Grand Prize* by Töscher et al. (2009) and *Combining Predictions for Accurate Recommender Systems* by Jahrer et al. (2010).

2.5.4 Different underlying datasets

So far, as shown in Algorithm 3 and Figure 1, the level-0 generalizers were always trained on the same dataset (named D). A priori, there is no reason why that should be the case. Actually, in *Principles of Forecasting*, Armstrong (2001) advocates for combining forecasts which draw from different sources of information.

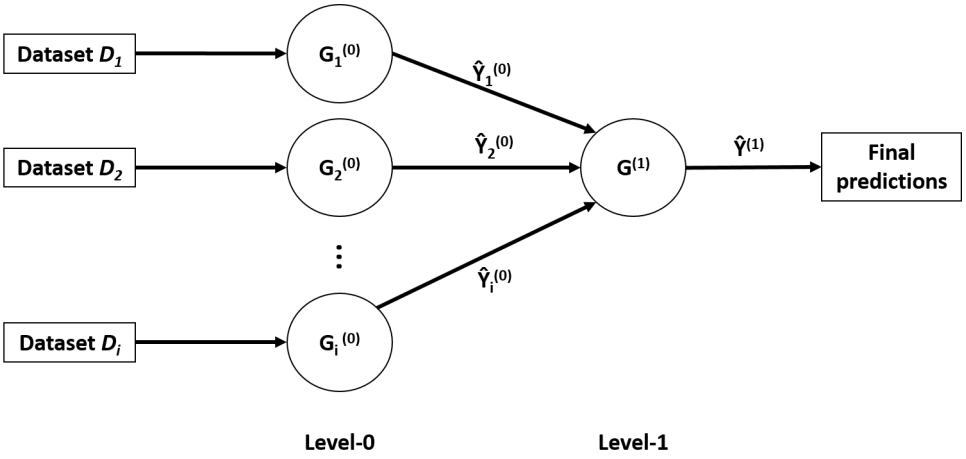


Figure 2: Schema of the stacked generalization process using different datasets

Given that future observations to score are available in all the formats of D_i, the $G_i^{(0)}$ could well be trained using different datasets. In this context, one could see *bagging* (Breiman, 1996a) as a special case of stacking. After all, bagging consists of creating I new datasets by doing bootstrap samples and then fitting I generalizers (this is level-0). Then, given a new observation to score, the $G_i^{(0)}$ generate a fitted value and the results are averaged (regression) or the majority vote wins (classification). This last step would be level-1, where in this case no actual learning is made.

The idea of generating bootstrap samples for level-0 and then fitting a model at level-1 to aggregate the results was studied by Ting and Witten (1997). They called the technique *bag-stacking*. Their research shows that "bag-stacking almost always has a higher predictive accuracy than bagging" (p. 367). Given what we have seen so far, it is not surprising to see that learning at level-1 (using, for example, a regularized linear regression) tends to yield better results than simply averaging the outputs of level-0 (or do a majority vote).

3 Stacked generalization for parameter selection

The second section of this paper is concerned with formulating an hypothesis, performing an experiment in order to test that hypothesis, analysing the results and drawing conclusions.

First, a formal hypothesis will be presented. Second, we will explain the process followed to test the hypothesis. Third, the dataset that will be used for the experiment will be presented. Fourth, the experiment will be performed and intermediary results will be shown along the way. Different learning schemes for level-1 will be tested: linear regressions, ridge and lasso regressions and linear regressions with constraints on the coefficients will be employed, as suggested by Breiman (1996b). We will also experiment with decision trees for reasons that will be explained later on.

Finally, we will summarize and analyse the results, confirm or reject the hypothesis and draw conclusions from the experiment.

3.1 Hypothesis

As mentioned in *Section 2.3.2 Unstable Model Selection*, Breiman (1996a) suggests that combining models could help when the process of selecting the single best model is unstable. This idea can logically be extended to parameter selection. Indeed, a number of learning schemes require to select a certain parameter, which can be done with cross-validation for example.

The hypothesis that will be tested here is as follows: if such parameter selection technique is unstable (i.e. suggests a different value when the data is reshuffled), then it can be beneficial to aggregate the fitted values obtained with the learning scheme using different parameters. By beneficial, we mean that this aggregation would result in lower errors than simply picking the underlying model with the "best" parameter. The criteria used to compare the techniques is the root mean squared error (RMSE).

In order to test if smaller RMSE scores can be obtained by combining the fitted values, the following experiment will be conducted.

3.2 Experiment: the process

The experiment will be conducted in two phases. First, a supervised-learning scheme requiring some parameter selection will be used on a dataset. Using cross-validation, the best parameter will be kept and the corresponding cross-validated RMSE score will be used as benchmark. The cross-validation will have to return unstable results (i.e. suggest different values for the parameter when the data is reshuffled).

Second, the fitted values of the different fitted models (using different parameters) will be used as inputs in level-1. Different methods will be explored for the level-1 generalizer. The goal of this experiment is to see if it is possible, using stacked generalization, to obtain a lower RMSE on the chosen dataset by combining the fitted values of different level-0 models that use the same learning scheme but with different parameters.

3.3 Data for the experiment

The dataset used for the experiment is a list of opinion polls measuring the support for Donald Trump in the 2016 Republican Primary race. The data was compiled by HuffingtonPost

(2016). Table 4 displays a sample of the dataset[2].

ID	pollster	day.of.year	Trump
1	Morning Consult	92	0.45
2	NBC/SurveyMonkey	88	0.45
3	IBD/TIPP	88	0.38
4	Ipsos/Reuters	87	0.44
⋮	⋮	⋮	⋮
84	NBC/SurveyMonkey	4	0.38
85	IBD/TIPP	4	0.34
86	FOX	4	0.35
87	Ipsos/Reuters	2	0.41

Table 4: First and last few rows of the HuffingtonPost (2016) dataset

The columns of interest are *day.of.year*, which is the explanatory variable $X^{(0)}$, and *Trump*, which is the response variable Y. The variable *day.of.year* represents the date in 2016 on which the survey started. In the dataset, those dates vary from 2 January 2016 to 1 April 2016. The response *Trump* represents the surveyed percentage of support for candidate Donald Trump for the 2016 Republican Primary. It varies between 25% and 53% over the given period of time. The raw data is represented in a scatter-plot in Figure 3.

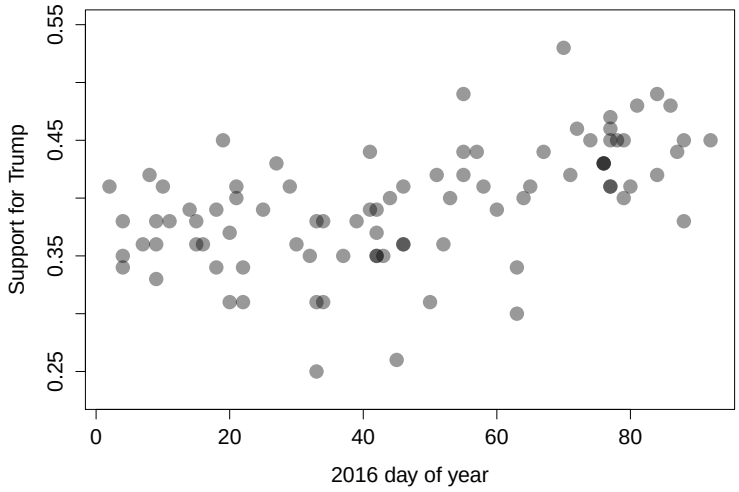


Figure 3: Raw data

3.4 Parameter selection: choosing a bandwidth

The task at hand is to apply a smoother to those polls in order to be able to "tell a story" from this noisy data. The chosen smoother for this task is a local linear regression. Fitting a local linear regression requires to select a parameter *bw*: the bandwidth. A typical approach

[2]See Table 13 in Appendix A (page 41) for the full dataset.

for choosing such parameter is to use cross-validation and select the bandwidth which returns the smallest root mean squared error (RMSE).

In the context of this dataset, given a ten-fold cross-validation, this means setting aside one tenth of the polls, fitting a local-linear regression using bandwidth bw on the remaining polls and generating fitted values for the polls previously set aside. Then this process is repeated for each fold and the RMSE for bandwidth bw is calculated. The goal is to find the bandwidth which will best describe the underlying trend (i.e. has the lowest RMSE) without over-fitting the data.

Figure 4 displays the results of the above process performed for bandwidths 2 to 30 (inclusively) on the dataset. According to those results, the lowest cross-validation score is obtained by choosing $bw = 14$, which returns a RMSE of 4.296%.

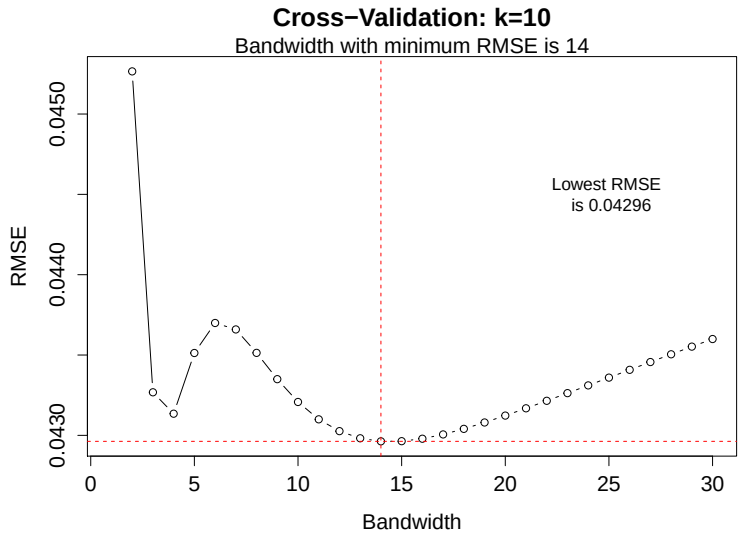


Figure 4: For each bandwidth between 2 and 30, calculate the 10-fold cross-validation score. In this case bandwidth 14 has the lowest score (0.04296).

One can easily notice from Figure 4 that multiple bandwidths have a quasi-equal RMSE to $bw = 14$. Indeed, bandwidths 13 to 16 have very similar scores, and $bw = 4$ is not too far either.

Since there is some randomness involved in the cross-validation process (when splitting the polls into 10 groups), a question should be asked: is the outcome of this cross-validation stable? The dataset being relatively small (87 observations), it seems reasonable to assume that the way the polls are split could influence the results. In order to see how robust the cross-validation is, the process has been performed 100 times and, for each of those 100 iterations, the "best" bandwidth bw was stored. Figure 5 shows a histogram where the height of the bars represents how many times the specified bandwidth was chosen as the best one.

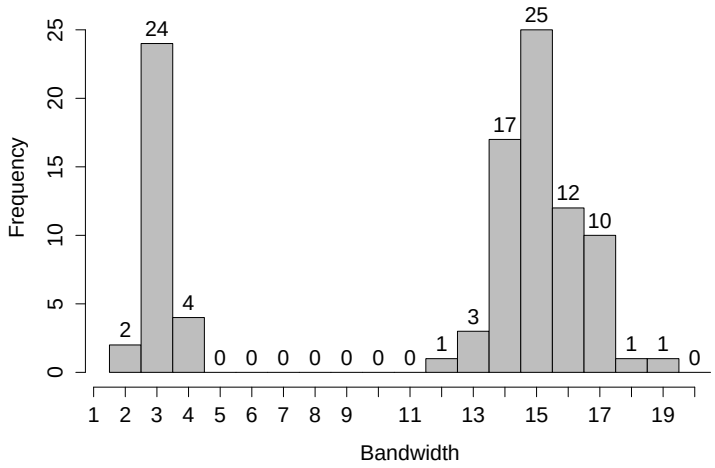


Figure 5: Repeating the cross-validation process 100 times with bandwidths 2 to 30 and selecting the best bandwidth at each iteration. Bandwidths between 20 and 30 inclusively are never selected.

Simply by looking at Figure 5, it's hard to make a solid argument for a clear winner. It's fair to say that the process leads to unstable conclusions. Indeed, bandwidths 3 (24%) and 15 (25%) seem to win approximately as many times, and overall there are four different bandwidths that win more than 10% of the time.

One key observation is that the winning bandwidths are clustered into two groups: one centered around $bw = 3$ and the other around $bw = 15$. If we fit a local linear regression using those two bandwidths on all the data and generate fitted values for *day.of.year* 2 to 92, we see that the curves tell quite a different story. Those curves, along with the original raw data, are shown on Figure 6.

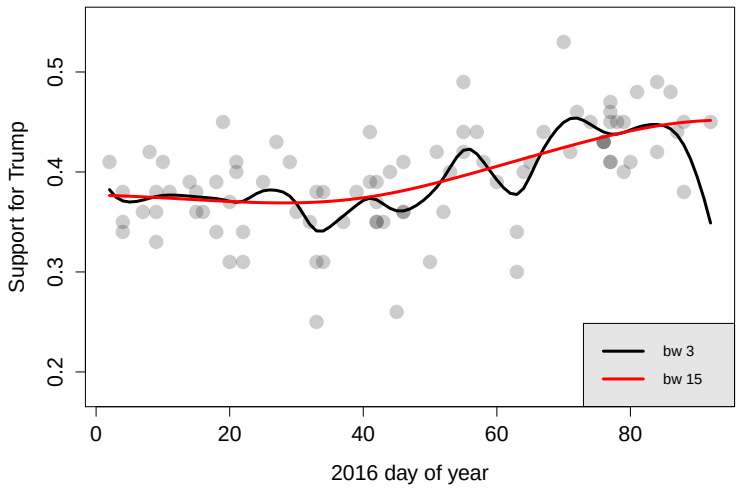
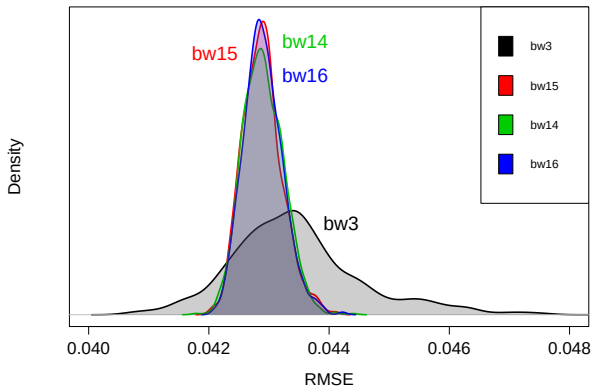


Figure 6: Comparing local linear regressions with bandwidths of 3 and 15

3.4.1 Comparing RMSEs of local linear regressions using different bandwidths

In order to get a better idea of what RMSE score we can expect from the different bandwidths, the following method is suggested: for each bandwidth of interest, do 1000 replications of generating fitted values through cross-validation and calculate the RMSE. Finally, display a density plot showing the distribution of RMSE scores for each bandwidth. Figure 7 shows the result of this process for bandwidths 3, 14, 15 and 16 (i.e. the ones which were winners more than 10% of the time).



Bandwidth	Mean	Standard Deviation
3	0.04351	0.00109
14	0.04289	0.00034
15	0.04289	0.00031
16	0.04291	0.00032

Figure 7: Comparing density of RMSE by bandwidth

Figure 7 shows that, as one can reasonably expect, bandwidths 14, 15 and 16 perform very similarly. Thus there is no need to duplicate all the future analysis for those three bandwidths; for the sections ahead, we will only work with $bw = 15$. Furthermore, we see that bandwidth 3 is much more unstable than $bw = 15$, as reflected by its standard deviation about three times higher than $bw = 15$. With regards to the means, they are very similar, with a slight advantage for $bw = 15$ (4.289%) over $bw = 3$ (4.351%).

3.4.2 Picking a winning bandwidth as benchmark

Because $bw = 15$ has a lower mean but $bw = 3$ has a larger standard deviation, it is hard to determine if the two bandwidths return significantly different results on average, statistically speaking. Consequently, formal statistical tests are required.

Because we have two independent samples of 1000 RMSE values and we want to know if one has a significantly smaller mean than the other, using a t-test seems appropriate. First, we have to determine if the two samples have similar variances. Looking at Figure 7, it's safe to assume that it is not going to be the case, but let's use the F-test to be sure.

F test to compare two variances

```
data:  RMSEs.bw15 and RMSEs.bw3
F = 0.081317, num df = 999, denom df = 999, p-value < 0.00000000000000022

alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.07182710 0.09206046

sample estimates:
ratio of variances
0.08131689
```

Table 5: F-test comparing the variances of the RMSEs generated with bandwidths 3 and 15

The p-value of the F-test (Table 5) is near zero, so we cannot assume equal variances. Indeed, the 95% confidence interval for the ratio of variances is [0.0718; 0.0921]. As a result, the Welch two sample t-test will be used to compare the means.

Welch Two Sample t-test

```
data:  RMSEs.bw15 and RMSEs.bw3
t = -16.958, df = 1160.4, p-value < 0.00000000000000022

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
-0.0006811038 -0.0005398434

sample estimates:
mean of x  mean of y
0.04289494 0.04350541
```

Table 6: Welch Two Sample t-test comparing the means of the RMSEs generated with bandwidths 3 and 15

The results of the Welch t-test (Table 6) show that at the 5% level, the mean of the two samples are significantly different. Indeed, the p-value of the test is near zero. The 95% confidence interval for the difference of the means is [0.054%; 0.068%]. In other words, at the 5% level, we can conclude that the expected RMSE of the local linear regression using a bandwidth of 3 is between 0.054% and 0.068% higher than when using a bandwidth of 15.

As a result, the benchmark for the "best" underlying model is the local linear regression using a bandwidth of 15.

3.5 Non-regularized linear stacking

The first learning scheme that will be used as level-1 generalizer is the ordinary least-square (OLS) linear regression. This implies that there is no restriction on the coefficients β: they can be negative or positive and they can sum to any number.

The process used to evaluate the performance of stacked generalization using OLS linear regression is Algorithm 3 (page 5). The final fitted values $\hat{Y}^{(1)}$ will be compared to the true

values Y and the root mean squared error will be calculated. Again, this will be done 1000 times so that we can have a distribution of RMSE values.

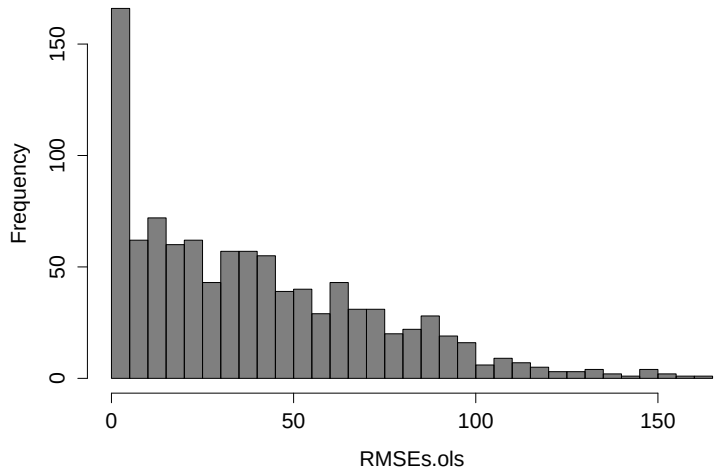


Figure 8: Histogram of RMSE obtained through 1000 replications of Algorithm 3 using OLS linear regression as $G^{(1)}$

As can be seen on Figure 8, the root mean squared errors seem to make little sense. Indeed, the middle 75% range of RMSEs is [224%; 8186%], which is way above the theoretical maximum error of 100%.

Table 7 (page 19) shows the coefficients calculated at each fold for one of the 1000 iterations. The striking result is that some coefficients are huge (sometimes in the billions), either positively or negatively. Consequently, there is no guarantee that the fitted values will stay inside the range of the data. For example, the final fitted value $\hat{Y}^{(1)}$ of observation *ID 87* is 682.04, which is obviously much higher than the theoretical maximum of 1 (and from the true value which is 0.41). Those very high (or low) β coefficients are likely the result of the fact that the inputs of level-1 are highly correlated (multi-collinearity).

It appears that in our situation, as Breiman (1996b) suggests, applying some kind of restriction on the coefficients so that the fitted values stay in the range of the data might be necessary.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
intercept	-0.01	-0.12	0.10	0.09	0.06	0.11	-0.07	0.00	0.03	0.21
fitted.bw2	-0.78	-1.08	-0.74	0.38	-0.67	0.21	2.66	-2.05	0.06	0.05
fitted.bw3	-4.51	-6.20	-14.37	-28.49	-6.03	-15.61	-37.82	-4.51	-23.39	-29.03
fitted.bw4	78.59	62.77	150.09	225.04	91.75	143.70	246.83	108.26	214.15	226.74
fitted.bw5	-534.08	-397.89	-824.82	-1068.49	-610.15	-786.63	-1062.78	-670.87	-1103.04	-1145.87
fitted.bw6	2110.61	1814.65	2933.33	3453.02	2419.11	2748.90	3327.02	2436.88	3695.56	3862.19
fitted.bw7	-4782.37	-4940.25	-6389.52	-7013.13	-5596.23	-5745.22	-7102.14	-5326.35	-7745.46	-7873.76
fitted.bw8	6357.09	8366.79	8416.22	8536.49	7709.39	7012.78	9824.45	7572.44	9872.25	9681.69
fitted.bw9	-5958.34	-9848.24	-7557.22	-7164.11	-7232.26	-5636.11	-9226.51	-9269.12	-8151.58	-9139.46
fitted.bw10	4320.08	6584.21	5037.06	5743.74	5024.66	3726.03	5730.49	9531.22	4748.57	8626.20
fitted.bw11	3403.10	4197.10	2585.43	-4.62	1918.07	1844.13	2733.65	961.28	2561.36	-35.32
fitted.bw12	-12380.79	-10453.30	-10284.80	-8516.30	-8904.04	-9144.54	-10047.51	-14185.73	-10599.23	-13231.96
fitted.bw13	-986.46	-2602.03	-1586.50	-1361.11	-3280.55	946.27	-4047.92	-1639.44	-1284.14	-1992.06
fitted.bw14	21779.20	12678.12	16567.40	18598.80	20348.61	14464.13	22894.52	24002.50	20911.43	29122.29
fitted.bw15	-7059.22	-4321.18	-2599.85	-7383.16	-7249.16	-6469.32	-8170.92	-5306.12	-6260.98	-7701.85
fitted.bw16	-9783.68	-5965.79	-7490.58	-2761.55	-5369.88	-5513.15	-4559.96	-8316.47	-9005.66	-13062.21
fitted.bw17	-4943.09	-7627.21	-4067.09	-7349.01	-4768.35	-3384.16	-4731.51	-6425.27	-6757.33	-8826.76
fitted.bw18	3406.72	-767.57	-797.32	-2015.59	358.68	-2862.11	-303.99	-245.02	432.14	-2929.00
fitted.bw19	11991.09	13381.53	6847.48	10755.97	6876.50	6487.66	1950.89	10899.92	12300.43	17862.85
fitted.bw20	4480.57	6645.96	9161.99	15975.72	5713.07	4080.77	10491.18	10403.78	11087.58	20182.25
fitted.bw21	5380.97	6983.05	-3766.63	2388.54	5140.52	16505.28	1712.09	-5064.49	4943.29	7426.42
fitted.bw22	15603.38	94360919.90	-7599.87	-38170.89	10626.69	-18080.81	35183.92	8394.37	-14639.87	-31839.51
fitted.bw23	-12785253.08	-846997332.73	520857.42	-24330915.72	-8891308.41	12578127.38	-19316299.84	-8927418.19	-24553684.28	-35159711.12
fitted.bw24	99139014.69	3428404583.52	-8526934.20	199133038.21	63840947.65	-100120207.28	143417786.78	59870086.69	196890437.37	282302993.46
fitted.bw25	-339348800.74	-8120529218.87	44547866.89	-711161385.12	-200725807.80	352425197.61	-468017736.30	-174700770.29	-688714575.49	-988660657.75
fitted.bw26	655107421.56	12240631567.90	-117026802.43	1428295243.27	353177266.35	-706460157.56	861601290.80	282210370.91	1353771827.85	1946536469.68
fitted.bw27	-765073046.92	-11967202668.47	175611188.20	-1734532544.62	-372385124.88	865441872.46	-961549607.66	-267835036.63	-1608514688.86	-2317921140.82
fitted.bw28	538312766.92	7381899075.56	-153360603.39	1269711788.02	233578431.66	-644354865.11	648180228.25	145859574.60	1152004251.41	1664745049.96
fitted.bw29	-210715738.16	-2618508633.47	72773700.11	-517583427.26	-80087744.04	268769338.49	-243691637.73	-40442895.85	-459516690.83	-666308482.52
fitted.bw30	35331158.69	407927924.41	-14537991.54	90485362.73	11470130.60	-48279627.52	39331170.27	3948234.53	78627927.62	114466295.70
Sum	1.03	1.07	0.85	0.85	0.92	0.78	1.14	0.98	0.96	0.69

Table 7: Coefficients of the 10 folds of one iteration of Algorithm 3 when using a **OLS linear regression** as the level-1 generalizer

3.6 Ridge and Lasso linear stacking

As Breiman (1996b) suggests, one way to deal with multi-collinearity is to use Ridge or Lasso regressions, which use a penalty term that pushes some coefficients towards zero. Sections 3.6.1 and 3.6.2 are dedicated to testing Algorithm 3 using those two methods on our 2016 Republican Primary dataset.

3.6.1 Ridge

The ordinary least square (OLS) defines its coefficients β as the values which minimize the sum of squared errors. Using the notation of Algorithm 3, and given a dataset containing M observations, β can be defined as

$$\beta = \arg\min_{\beta} \left\{ \sum_{m=1}^{M} (Y_m - X_m\beta)^2 \right\} \tag{5}$$

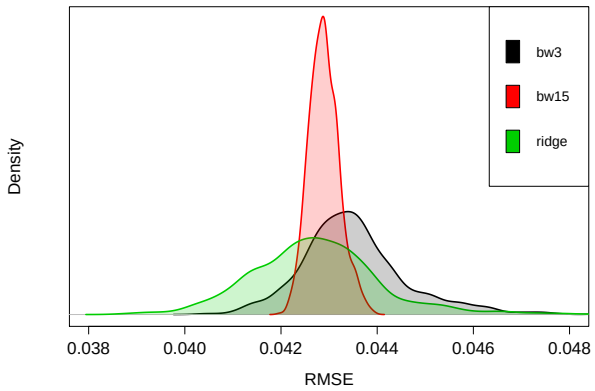
Similarly, the problem can be expressed as an optimization one, as in Equation 6. This equation will be revisited later on when we address regularized linear stacking.

$$\text{Minimize} \qquad \|X\beta - Y\|^2 \tag{6}$$

In a ridge regression, the ridge penalty term is added to the sum of squared errors. This term makes sure to penalize very large (squared) coefficients. Furthermore, a shrinkage parameter λ is added to adjust how important the penalty is. Thus, a λ value of zero would result in a normal OLS regression. Equation 7 shows how β is defined when using the ridge penalty.

$$\beta = \arg\min_{\beta} \left\{ \sum_{m=1}^{M} (Y_m - X_m\beta)^2 \quad + \quad \lambda \sum_{i=1}^{I} (\beta_i)^2 \right\} \tag{7}$$

Using a ridge regression as the level-1 generalizer $G^{(1)}$ should help fix the problem of very high (or low) coefficients encountered in *Section 3.5 Non-regularized linear stacking*. Once again, Algorithm 3 will be replicated 1000 times and the RMSE stored at each iteration.



	Mean RMSE	Standard Deviation
bw3	0.04351	0.00109
bw15	0.04289	0.00031
ridge	0.04272	0.00134

Figure 9: Comparing density of RMSE when using $bw = 3$, $bw = 15$ and ridge

Figure 9 shows the result of the 1000 iterations[3] of Algorithm 3 using a ridge regression as $G^{(1)}$ and compares those results with the ones obtained by simply using bandwidths of 3 or 15 at level-0. The first positive outcome of this experiment is that, as expected, the fitted values $\hat{Y}^{(1)}$ now seem to be closer to the true values Y than when using ordinary least square regressions, as reflected by the more "decent" RMSE values.

The second (and most important) outcome of this experiment is that the average RMSE of the stacked regression (4.272%) is smaller than the benchmark of 4.289%, which was obtained with $bw = 15$. That being said, one can clearly see on Figure 9 that the standard deviations of the RMSE values of *bw15* and *ridge* differ significantly. Thus, a formal statistical test should be used to determine if the means are truly different, statistically speaking.

The full outputs of the F-test and the Welch t-test can be found in Appendix C.1. At the 5% significance level, we can conclude that the expected RMSE of the stacked generalization using a ridge regression is significantly lower than the expected RMSE when simply using a local linear regression with bandwidth 15. More precisely, the 95% confidence interval for the difference of mean RMSE is [0.0089%; 0.0260%]. The p-value for this test is near zero. The Welch t-test was used because the F-test showed that equal variances cannot be assumed.

Table 8 shows the coefficients of the ridge regression for each of the ten folds of one iteration of Algorithm 3. As expected, the ridge penalty term shrinks the coefficients towards zero, which prevents the final fitted values $\hat{Y}^{(1)}$ from being very far away from the range of the data. None of the coefficients, however, is exactly equal to zero.

Another interesting point to note is that some coefficients are negative, which makes it difficult to interpret their relative importance in the final calculation of the fitted values. While a ridge regression shrinks the coefficients without setting any of them to zero, a lasso regression sets some coefficients *exactly* to zero. This learning scheme will be used as the level-1 generalizer in the Section 3.6.2.

[3]Note that at each of the 1000 replications, λ is determined through cross-validation. The mean λ is 0.14214. The distribution of λs can be seen on Figure 23 in Appendix D.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
intercept	-0.011	-0.005	-0.068	-0.017	-0.032	-0.012	0.041	-0.010	-0.003	-0.017
fitted.bw2	0.020	-0.002	0.092	0.058	0.045	0.017	0.289	-0.011	0.041	0.071
fitted.bw3	0.219	0.186	0.260	0.289	0.260	0.284	0.424	0.273	0.277	0.248
fitted.bw4	0.239	0.034	0.126	0.133	0.092	0.147	0.011	0.216	0.147	0.055
fitted.bw5	-0.024	-0.161	-0.173	-0.216	-0.200	-0.180	-0.284	-0.091	-0.175	-0.245
fitted.bw6	-0.237	-0.231	-0.352	-0.407	-0.343	-0.359	-0.351	-0.274	-0.356	-0.377
fitted.bw7	-0.284	-0.177	-0.348	-0.382	-0.306	-0.337	-0.281	-0.260	-0.334	-0.327
fitted.bw8	-0.237	-0.076	-0.256	-0.259	-0.195	-0.223	-0.165	-0.166	-0.224	-0.205
fitted.bw9	-0.163	0.020	-0.149	-0.129	-0.085	-0.101	-0.057	-0.073	-0.107	-0.084
fitted.bw10	-0.089	0.091	-0.054	-0.022	0.004	-0.002	0.021	0.002	-0.013	0.007
fitted.bw11	-0.024	0.138	0.023	0.057	0.071	0.070	0.068	0.057	0.058	0.068
fitted.bw12	0.031	0.168	0.082	0.114	0.122	0.120	0.094	0.100	0.109	0.106
fitted.bw13	0.075	0.186	0.127	0.156	0.162	0.156	0.108	0.132	0.147	0.131
fitted.bw14	0.107	0.193	0.156	0.182	0.188	0.178	0.111	0.153	0.171	0.145
fitted.bw15	0.131	0.195	0.177	0.199	0.206	0.192	0.109	0.166	0.186	0.152
fitted.bw16	0.147	0.190	0.190	0.208	0.216	0.198	0.102	0.171	0.193	0.153
fitted.bw17	0.157	0.181	0.198	0.210	0.218	0.198	0.090	0.169	0.193	0.149
fitted.bw18	0.163	0.167	0.201	0.207	0.214	0.192	0.076	0.161	0.188	0.141
fitted.bw19	0.164	0.148	0.200	0.197	0.202	0.181	0.061	0.147	0.175	0.129
fitted.bw20	0.162	0.125	0.195	0.182	0.184	0.163	0.047	0.128	0.157	0.114
fitted.bw21	0.154	0.097	0.183	0.160	0.158	0.139	0.037	0.104	0.133	0.097
fitted.bw22	0.141	0.066	0.165	0.132	0.125	0.111	0.031	0.077	0.105	0.080
fitted.bw23	0.123	0.032	0.141	0.100	0.088	0.079	0.030	0.049	0.074	0.065
fitted.bw24	0.099	-0.002	0.110	0.065	0.049	0.046	0.033	0.022	0.044	0.054
fitted.bw25	0.071	-0.034	0.074	0.031	0.009	0.014	0.038	-0.002	0.015	0.048
fitted.bw26	0.039	-0.064	0.035	-0.001	-0.028	-0.014	0.045	-0.022	-0.009	0.047
fitted.bw27	0.005	-0.089	-0.005	-0.029	-0.061	-0.038	0.050	-0.038	-0.029	0.048
fitted.bw28	-0.029	-0.109	-0.043	-0.052	-0.089	-0.057	0.053	-0.050	-0.045	0.052
fitted.bw29	-0.061	-0.125	-0.078	-0.069	-0.112	-0.071	0.054	-0.059	-0.056	0.055
fitted.bw30	-0.090	-0.138	-0.109	-0.083	-0.130	-0.083	0.052	-0.067	-0.066	0.056
Sum	0.998	1.004	1.100	1.014	1.032	1.008	0.937	1.004	0.996	1.016

Table 8: Coefficients of the 10 folds of one iteration of Algorithm 3 when using a **Ridge** regression at level-1

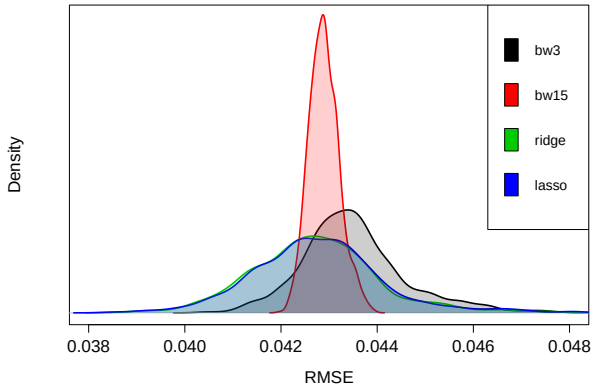
3.6.2 Lasso

As mentioned earlier, Lasso regressions behave in a very similar way as ridge regressions. Equation 8 shows how it finds the coefficients β. As one can notice, the only difference with Equation 7 is that the penalty term includes the absolute values of the βs, and not the squared values.

$$\beta = \arg\min_{\beta} \left\{ \sum_{m=1}^{M} (Y_m - X_m\beta)^2 \quad + \quad \lambda \sum_{i=1}^{I} |\beta_i| \right\} \tag{8}$$

Because of the way the penalty term is designed, a lasso regression "tends to produce some coefficients that are exactly 0 and hence gives interpretable models" (Tibshirani, 1996, p. 267). More specifically, it sets the coefficients of the variables which are highly correlated with other variables to zero. As mentioned in *Section 2.4: Known issues*, multi-collinearity is a problem that has to be addressed when using stacked generalization, especially so when all the level-1 inputs are derived from the same learning scheme at level-0. Thus it seems reasonable to assume that, like ridge regressions, lasso regressions should perform better than OLS in stacked generalization.

Figure 10 shows the result of 1000 iterations[4] of Algorithm 3 using lasso as $G^{(1)}$, along with the previous results of *bw3*, *bw5* and *ridge* as reference.



	Mean RMSE	Standard Deviation
bw3	0.04351	0.00109
bw15	0.04289	0.00031
ridge	0.04272	0.00134
lasso	0.04277	0.00140

Figure 10: Comparing density of *bw3*, *bw15*, *ridge* and *lasso*

It appears that *ridge* and *lasso* have an almost indistinguishable distribution of RMSE. In fact, a two-sample t-test shows that we **cannot** conclude, at the 5% level, that the mean RMSE of *ridge* and *lasso* are significantly different. The p-value of this test is 0.4022. The two sample t-test was used (as opposed to the Welch t-test) because the F-test showed that equal variances can be assumed. The full output of those tests can be found in Appendix C.2.

A possible explanation for the similitude of the RMSEs is as follows: when looking at the values of λ more closely, we realize that the λs of *ridge*, much higher than the ones of *lasso*,

[4]Note that once again, at each of the 1000 replications, λ is determined through cross-validation. The mean λ is 0.00333. The distribution of λs can be seen on Figure 24 in Appendix D.

"compensate" for the smaller sum of squared coefficients. Indeed, in the ridge penalty, the coefficients are squared; since they are all in the range [-1;1], when squared the values get lower. In the lasso penalty term, however, only the absolute value of the coefficients is used. As a result we have

ridge penalty:	high λ	$\times$	low value
lasso penalty:	low λ	$\times$	high value

which might explain why *ridge* and *lasso* have very similar distributions of RMSEs.

Table 9 shows the coefficients of the lasso regression for each of the ten folds of one iteration of Algorithm 3. As expected, the lasso penalty term pushes most coefficients to exactly zero. Indeed, the folds only keep between 4 and 7 non-zero coefficients out of 29 bandwidths. Interestingly, for the vast majority of folds, the most important non-negative coefficients are $bw = 3$ and a combination of coefficients between $bw = 14$ and $bw = 18$, which is consistent with the frequencies of "best bandwidths" shown on Figure 5 (page 15). It is also consistent with Breiman's (1996b) conlusion that the biggest gains of stacking come from combining dissimilar fitted values. Figure 6 (page 15) showed that local linear regressions with bandwidths 3 and 15 tell quite different stories, the former being more sensible to local variations in the data while the later is more descriptive of the longer trend.

In conclusion, so far this experiment seems to confirm our hypothesis that when using a learning scheme which requires the selection of some parameter and that this selection process is unstable, it can be beneficial to aggregate the fitted values obtained with different parameters instead of simply selecting the best parameter.

It should also be noted that the lasso penalty term doesn't prevent the coefficients from being negative. Consequently, it can be difficult to measure and interpret the relative importance of the bandwidths at level-1. A possible solution to this problem was suggested by Breiman (1996b): combine the coefficients linearly while applying some restrictions (regularizations) on the coefficients, thus making the interpretation more straightforward. The next section will be dedicated to such methods. More specifically, we will try to see if it is possible to linearly combine the level-0 fitted values $\hat{Y}^{(0)}$ in such a way that it is possible to evaluate the relative importance of the bandwidths while still expecting lower errors than our benchmark RMSE.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
intercept	0.024	-0.015	-0.025	0.064	0.020	0.010	-0.026	-0.026	-0.032	-0.049
fitted.bw2	0	0	0	0	0	0	0	0	0	0
fitted.bw3	0.698	0.644	0.707	1.442	0.630	0.246	0.735	0.726	0.676	0.732
fitted.bw4	0	0	0	0	0	0	0	0	0	0
fitted.bw5	0	0	0	-1.800	0	0	0	0	0	0
fitted.bw6	-1.166	-1.406	-1.635	0	-1.348	-0.387	-1.711	-1.614	-1.761	-1.618
fitted.bw7	0	0	0	0	0	0	0	0	0	0
fitted.bw8	0	0	0	0	0	0	0	0	0	0
fitted.bw9	0	0	0	0	0	0	0	0	0	0
fitted.bw10	0	0	0	0	0	0	0	0	0	0
fitted.bw11	0	0	0	0	0	0	0	0	0	0
fitted.bw12	0	0	0	0.805	0	0	0	0	0	0
fitted.bw13	0	0	0	0.368	0	0	0	0	0	0
fitted.bw14	0	0	0	0.013	0	0.654	1.065	1.205	1.038	0
fitted.bw15	0.646	0.944	0.808	0	0.758	0.362	0.691	0.664	0.711	1.392
fitted.bw16	0.466	0.579	0.574	0	0.510	0.096	0.269	0.072	0.355	0.608
fitted.bw17	0.228	0.248	0.344	0	0.278	0	0	0	0.048	0
fitted.bw18	0.064	0.018	0.215	0	0.114	0	0	0	0	0
fitted.bw19	0	0	0.037	0	0	0	0	0	0	0
fitted.bw20	0	0	0	0	0	0	0	0	0	0
fitted.bw21	0	0	0	0	0	0	0	0	0	0
fitted.bw22	0	0	0	0	0	0	0	0	0	0
fitted.bw23	0	0	0	0	0	0	0	0	0	0
fitted.bw24	0	0	0	0	0	0	0	0	0	0
fitted.bw25	0	0	0	0	0	0	0	0	0	0
fitted.bw26	0	0	0	0	0	0	0	0	0	0
fitted.bw27	0	0	0	0	0	0	0	0	0	0
fitted.bw28	0	0	0	0	0	0	0	0	0	0
fitted.bw29	0	0	0	0	0	0	0	0	0	0
fitted.bw30	0	0	0	0	0	0	0	0	0	0
Sum	0.960	1.012	1.025	0.892	0.962	0.981	1.023	1.027	1.035	1.065

Table 9: Coefficients of the ten folds of one iteration of Algorithm 3 using a **LASSO** regression

3.7 Regularized linear stacking

In this section, we use linear regressions as level-1 generalizers, but with some constraints on the coefficients. Because of those constraints, the problem becomes an optimization one and can be expressed as follows:

$$\begin{aligned} \text{Minimize} \quad & \|X\beta - Y\|^2 \\ \text{subject to} \quad & I\beta \geqslant h \\ & E\beta = f \end{aligned} \tag{9}$$

where

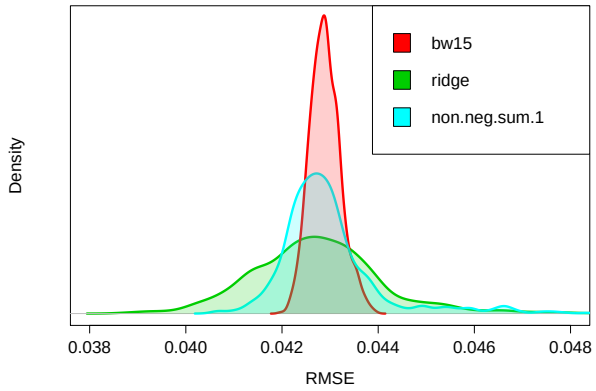
- X is the matrix of level-0 fitted values $\hat{Y}^{(0)} = X^{(1)}$
- β is a column matrix containing the coefficients we are loking for
- Y is the vector of true values
- I is the identity matrix with same length as β
- h is a column matrix of same length as β containing only zeros
- E is a row matrix of same length as β containing only ones
- $f = 1$

The first constraint $I\beta \geqslant h$ makes sure that all coefficients are greater or equal to zero. Note that because I is the identity matrix, $I\beta \equiv \beta$. However we still write $I\beta$ instead of simply β because the R package used to solve Equation 9 requires such notation. The second constraint $E\beta = f$ makes sure that the sum of all coefficients is equal to one. Since both Breiman (1996b) and Ting and Witten (1999) argue that the sum-to-one constraint is largely unnecessary, at first we will use the two constraints, and then only the non-negativity constraint. We will then be able to compare the results of both approaches.

Note that the solution to the *least squares with equalities and inequalities* problem shown at Equation 9 was originally developed by Lawson and Hanson (1995). For this experimentation, the R package *limSolve* was used, which itself calls another package (*quadprog*) which contains routines to solve quadratic programming problems. See Appendix B for the actual code.

3.7.1 Non-negative coefficients and sum-to-one constraints

We start by applying the two constraints to the problem. Figure 11 shows a density plot of the resulting RMSEs of 1000 replications of Algorithm 3 using Equation 9 as the level-1 generalizer. *Ridge* and *bw15* are shown as reference. Even though the density plot seems to show that the mean of *non.neg.sum.1* (4.296%) is smaller than the mean of *bw15* (4.289%), it is actually slightly higher because of the long tail of *non.neg.sum.1*'s distribution. This is reflected in the standard deviation that is about three times bigger (0.031% versus 0.104%).



	Mean RMSE	Standard Deviation
bw15	0.04289	0.00031
ridge	0.04272	0.00134
non.neg.sum.1	0.04296	0.00104

Figure 11: Comparing density of *bw15*, *ridge* and *non.neg.sum.1*

A Welch two sample t-test comparing the mean RMSE of *non.neg.sum.1* and *bw15* shows that, at the 5% level, we cannot conclude that the difference between the two is significant. Indeed, the p-value for this test is 6.01%. Note that the Welch test was used because the F-test showed that equal variances cannot be assumed. See Appendix C.3 for the full outputs of the tests.

Table 10 shows the coefficients of the ten folds for one iteration of Algorithm 3. It is interesting to note that for each fold, only 2 or 3 bandwidths have non-zero coefficients. A clear pattern seems to emerge: the best combination is one which combines the ultra-local (*bw=2*) and longer-term (*bw=18+*) trends. This is consistent with Breiman (1996b) and Ting & Witten's (1999) assumption that with stacked generalization, the biggest gains come from aggregating dissimilar predictors. However, for our problem and dataset, restricting the coefficients with both constraints (non-negativity and sum-to-one) doesn't seem to significantly improve the average RMSE (when compared to bw15), nor does it make it more stable (the standard variation is about three times higher).

Some authors (Breiman, 1996b; Ting and Witten, 1999) argue that the sum-to-one constraint is unnecessary. In the next section, we will see if loosening the quadratic programming problem will lead to lower, higher or stable RMSEs.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
fitted.bw2	0.313	0.517	0.438	0.416	0.456	0.427	0.376	0.385	0.409	0.419
fitted.bw3	0	0	0	0	0	0	0	0	0	0
fitted.bw4	0	0	0	0	0	0	0	0	0	0
fitted.bw5	0	0	0	0	0	0	0	0	0	0
fitted.bw6	0	0	0	0	0	0	0	0	0	0
fitted.bw7	0	0	0	0	0	0	0	0	0	0
fitted.bw8	0	0	0	0	0	0	0	0	0	0
fitted.bw9	0	0	0	0	0	0	0	0	0	0
fitted.bw10	0	0	0	0	0	0	0	0	0	0
fitted.bw11	0	0	0	0	0	0	0	0	0	0
fitted.bw12	0	0	0	0	0	0	0	0	0	0
fitted.bw13	0	0	0	0	0	0	0	0	0	0
fitted.bw14	0	0	0	0	0	0	0	0	0	0
fitted.bw15	0	0	0	0	0	0	0	0	0	0
fitted.bw16	0	0	0	0	0	0	0	0	0	0
fitted.bw17	0	0	0	0	0	0	0	0	0	0
fitted.bw18	0	0	0	0	0	0	0.624	0	0	0
fitted.bw19	0	0	0	0	0	0	0	0	0	0
fitted.bw20	0	0	0	0.584	0	0	0	0.312	0	0
fitted.bw21	0	0	0.078	0	0	0	0	0.303	0.591	0
fitted.bw22	0	0	0.484	0	0	0	0	0	0	0.581
fitted.bw23	0	0	0	0	0	0	0	0	0	0
fitted.bw24	0	0.215	0	0	0	0	0	0	0	0
fitted.bw25	0	0.268	0	0	0.544	0	0	0	0	0
fitted.bw26	0.459	0	0	0	0	0	0	0	0	0
fitted.bw27	0.228	0	0	0	0	0.573	0	0	0	0
fitted.bw28	0	0	0	0	0	0	0	0	0	0
fitted.bw29	0	0	0	0	0	0	0	0	0	0
fitted.bw30	0	0	0	0	0	0	0	0	0	0
Sum	1	1	1	1	1	1	1	1	1	1

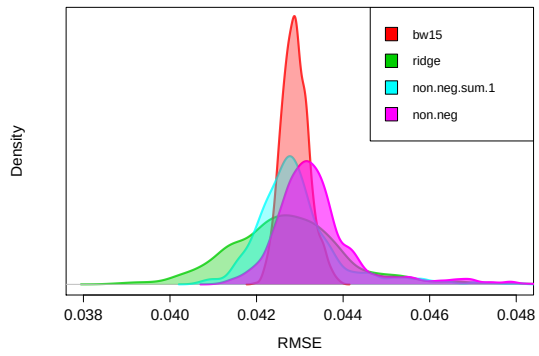
Table 10: One iteration of Algorithm 3: coefficients of the ten folds when using **non-negativity and sum-to-one** constraints

3.7.2 Non-negative coefficients constraint only

In this subsection, Equation 9 is used as the level-1 generalizer, with the adjustment that the second constraint (coefficients sum to one) is dropped:

$$\begin{aligned} \text{Minimize} \quad & \|X\beta - Y\|^2 \\ \text{subject to} \quad & I\beta \geqslant h \end{aligned} \tag{10}$$

Once again, 1000 replications of Algorithm 3 are executed and the resulting RMSE of each iteration is stored in *non.neg*. Figure 12 shows a density plot of *non.neg*, along with *non.neg.sum.1*, *ridge* and *bw15* as reference.



	Mean RMSE	Standard Deviation
bw15	0.04289	0.00031
ridge	0.04272	0.00134
non.neg.sum.1	0.04296	0.00104
non.neg	0.04338	0.00104

Figure 12: Comparing density of RMSE when using bandwidth 15, ridge regression, regularized linear combinations using non-negativity and sum-to-one constraints (cyan) and non-negativity only (purple)

It appears that *non.neg.sum.1* and *non.neg* return similar results. Indeed, the standard deviations are almost identical (0.104%), and the mean RMSE of *non.neg* (4.338%) is slightly higher than the mean RMSE of *non.neg.sum.1* (4.296%). At the 5% level, this difference is significant: the p-value of the t-test is near zero and the 95% confidence interval for the difference of means is [0.035%;0.053%]. The full output of the test is in Appendix C.4.

At first those results may seem surprising. We expect that by removing a constraint to a minimization problem, we would get a lower (or at least equal) objective. One possible explanation for the higher RMSEs is that while the minimization problem at Equation 10 finds a smaller sum of squared errors at *step 12*[5] of Algorithm 3, it might slightly overfit the data and hence generalizes less well on the new data at *step 13*[6]. By imposing the sum-to-one constraint like in Equation 9, the fitted model $G^{(1)}$ might fit the training data a bit less accurately, but generalizes better when new data is presented. This is only an assumption, but it appears to be a plausible one.

Table 11 shows the coefficients of the ten folds for one iteration of Algorithm 3 using Equation 10 at level-1. One can see that the coefficients are very similar to the ones generated with both constraints (Table 10). The sum of coefficients tends to stay very close to one without any specific constraint on the sum of coefficients.

[5]Fit $G^{(1)}$ on $\delta^{[-k]}$

[6]Use $G^{(1)}$ to generate fitted values for $\delta^{[k]}$

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
fitted.bw2	0.325	0.523	0.424	0.397	0.456	0.410	0.341	0.374	0.396	0.403
fitted.bw3	0	0	0	0	0	0	0	0	0	0
fitted.bw4	0	0	0	0	0	0	0	0	0	0
fitted.bw5	0	0	0	0	0	0	0	0	0	0
fitted.bw6	0	0	0	0	0	0	0	0	0	0
fitted.bw7	0	0	0	0	0	0	0	0	0	0
fitted.bw8	0	0	0	0	0	0	0	0	0	0
fitted.bw9	0	0	0	0	0	0	0	0	0	0
fitted.bw10	0	0	0	0	0	0	0	0	0	0
fitted.bw11	0	0	0	0	0	0	0	0	0	0
fitted.bw12	0	0	0	0	0	0	0	0	0	0
fitted.bw13	0	0	0	0	0	0	0	0	0	0
fitted.bw14	0	0	0	0	0	0	0	0	0	0
fitted.bw15	0	0	0	0	0	0	0	0	0	0
fitted.bw16	0	0	0	0	0	0	0	0	0	0
fitted.bw17	0	0	0	0	0	0	0	0	0	0
fitted.bw18	0	0	0	0	0	0	0	0	0	0
fitted.bw19	0	0	0	0	0	0	0.648	0	0	0
fitted.bw20	0	0	0	0.099	0	0	0	0	0	0
fitted.bw21	0	0	0.074	0.497	0	0	0	0.622	0.601	0
fitted.bw22	0	0	0.498	0	0	0	0	0	0	0.592
fitted.bw23	0	0	0	0	0	0	0	0	0	0
fitted.bw24	0	0	0	0	0.005	0	0	0	0	0
fitted.bw25	0	0.479	0	0	0.540	0.049	0	0	0	0
fitted.bw26	0	0	0	0	0	0.536	0	0	0	0
fitted.bw27	0.259	0	0	0	0	0	0	0	0	0
fitted.bw28	0.420	0	0	0	0	0	0	0	0	0
fitted.bw29	0	0	0	0	0	0	0	0	0	0
fitted.bw30	0	0	0	0	0	0	0	0	0	0
Sum	1.004	1.002	0.996	0.993	1.001	0.995	0.989	0.996	0.997	0.995

Table 11: **Non-negativity** constraint only: coefficients of the 10 folds of one iteration of Algorithm 3

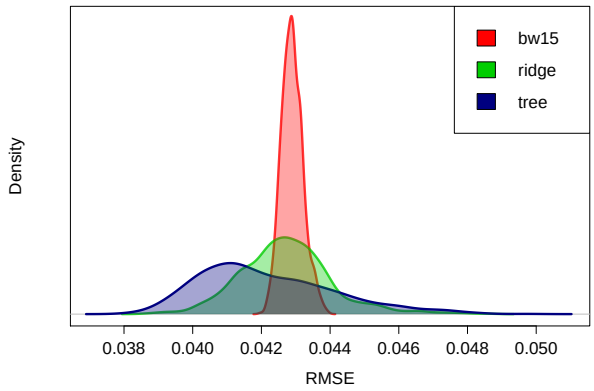
3.8 Non-linear stacking: regression trees

So far we only dealt with linear models at level-1. While (some of) those methods have the advantage of returning easy-to-interpret βs, one downside of the generalizers used in the previous sections is that the βs are constant throughout the range of $X^{(1)}$.

In the context of our problem (combining fitted values obtained by using different parameters), this means such approaches are not able to discern whether a bandwidth works best on some region of $X^{(0)}$ while other bandwidths might work best on other regions. In other words, the linear methods used so far "settle" for the linear combination of fitted values which works best overall, not locally.

One potential way around this problem is to use regression trees (as defined by Breiman et al. (1984)) at level-1. It seems fair to assume that a more dynamic technique at level-1 could improve the accuracy of the stacked generalization process. The hope is that the tree will be able to learn which fitted values can be trusted and, especially, where.

Similarly to the previous section, 1000 replications of Algorithm 3 using a regression tree as $G^{(1)}$ will be performed and the RMSE of each iteration will be stored. For each iteration, at each fold, a tree is fully grown and then pruned. The final depth of the tree is determined through cross-validation. Figure 13 shows a density plot of the resulting RMSEs and compares those results with the ones obtained by simply using a bandwidth of 15 at level-0 and by using a ridge regression at level-1 (the best performing generalizer thus far).



	Mean RMSE	Standard Deviation
bw15	0.04289	0.00031
ridge	0.04272	0.00134
tree	0.04216	0.00195

Figure 13: Comparing density of RMSE when using bandwidth 15, ridge regression and a decision tree

One can see from Figure 13 that overall, is seems like regression trees return smaller RMSEs than ridge regressions. Indeed, the average of *tree* (4.216%) is slightly smaller than *ridge* (4.272%). Statistical tests show that, at the 5% level, the mean of *tree* is significantly lower than both the means of *bw15* and *ridge* (see Appendix C.5 and C.6 respectively for the full outputs of the tests).

Figure 14 shows the tree grown at each of the ten folds of one iteration of Algorithm 3. For that particular iteration, 7 of the 10 trees are based solely on the fitted values obtained with a bandwidth of 3, while the remaining trees use a combination of multiple fitted values.

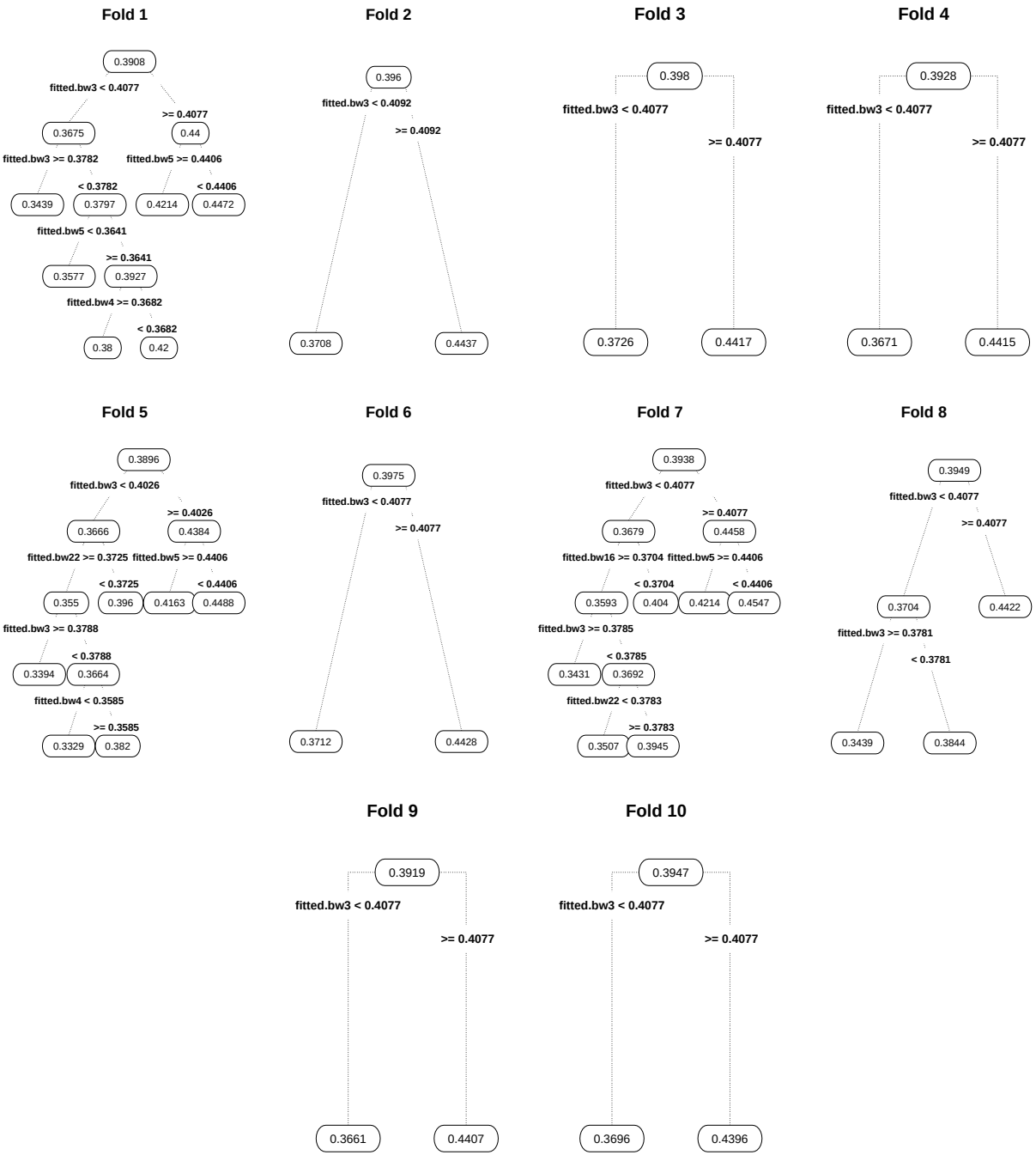


Figure 14: **Regression trees** fit at each fold of one iteration of Algorithm 3

Our goal seems to be partially reached here. First, the mean RMSE is significantly lower than the best performing method used to far, which is the ultimate goal. On the other hand, we are also trying to show that learning at level-1 using *multiple* sets of fitted values can lead to lower errors. Consequently, since only a few trees of Figure 14 end up using multiple sets of fitted values, we cannot conclude that "combining" multiple bandwidths with decision trees necessarily leads to better results. However, at least for this dataset and these generalizers, we can conclude that, on average, learning at the meta level (level-1) does lead to lower error rates.

In the next subsection, we will look at what the final fitted values of Algorithm 3 using regression trees might look like on a plot and we will compare those results with the other techniques.

3.9 Summarizing and analysing the results

The present section began with the following hypothesis: if a parameter selection technique such as cross-validation returns unstable results, then it can be beneficial to aggregate the fitted values of the different models rather than simply picking the "best" one. Given the results presented in the previous sections, we can conclude that it is possible to obtain a significantly lower average RMSE when stacking fitted values derived from using different parameters, than when simply picking the best parameter.

For the current data, and using local linear regressions at level-0, we saw that the best linear generalizer to use at level-1 is a ridge (or lasso) regression. Because with such technique there is no constraint on the βs, they can have negative values, which makes the interpretation of their relative importance more difficult. Linear stacking with restrictions on the coefficients does produce more interpretable coefficients, but doesn't generate a significantly lower average error rate than the initial benchmark.

Furthermore, we saw that on average, using more dynamic approaches at level-1 such as regression trees can lead to even significantly lower errors than ridge regressions. Table 12 shows a summary of the 1000 RMSEs generated with all techniques.

	bw3	bw15	ridge	lasso	non.neg.sum.1	non.neg	tree
Min.	0.04037	0.04198	0.03873	0.03848	0.04062	0.04115	**0.03822**
1st Q	0.04282	0.04268	0.04189	0.04192	0.04237	0.04277	**0.04072**
Median	0.04340	0.04288	0.04269	0.04272	0.04279	0.04321	**0.04182**
Mean	0.04351	0.04289	0.04272	0.04277	0.04296	0.04338	**0.04216**
3rd Q	0.04401	**0.04310**	0.04343	0.04347	0.04323	0.04366	0.04337
Max.	0.04858	**0.04394**	0.04856	0.04974	0.04907	0.04925	0.04971
Std Dev.	0.00109	**0.00031**	0.00134	0.00140	0.00099	0.00104	0.00195

Table 12: Comparing density of RMSE of all stacking techniques and *bw3* and *15* as reference. For all rows, the smallest value is in bold.

As Table 12 shows, the regression tree is the generalizer which returns the lowest average root mean squared error. However it is worth mentioning that the 1000 RMSEs generated with this technique have the largest standard deviation of all the techniques used. This

means that on average, regression trees will perform significantly better than the other techniques, but when compared with the benchmark *bw15*, they are also more prone to large errors. It is unclear if this conclusion is valid only for this dataset, or in general.

Figures 15 and 16 show visual representations of the RMSEs which illustrate well the lower mean and higher variance of *tree* versus the other techniques. One can clearly see on Figure 15 that the density curve of *tree* is more flat than the other curves, but also that its peak is centered at ∼4.1% while the other peaks are centered at ∼4.3%. However, because of its long tail, *tree*'s mean is only 0.073% lower than the mean of *bw15*, which is still significant at the 5% level.

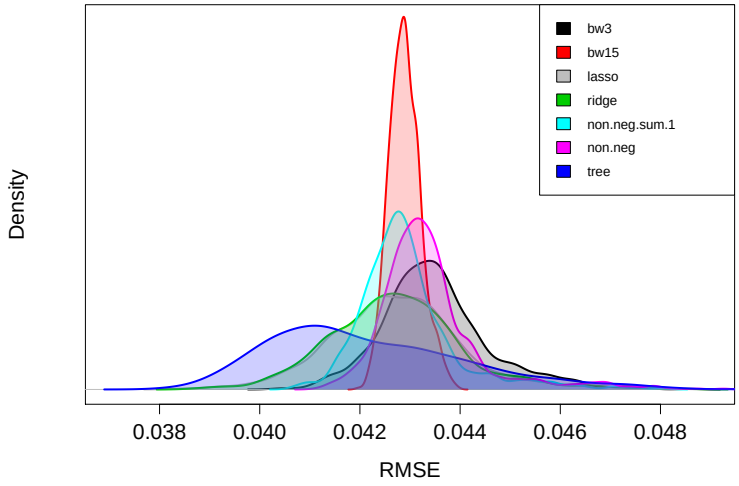


Figure 15: Comparing density of RMSE of all techniques

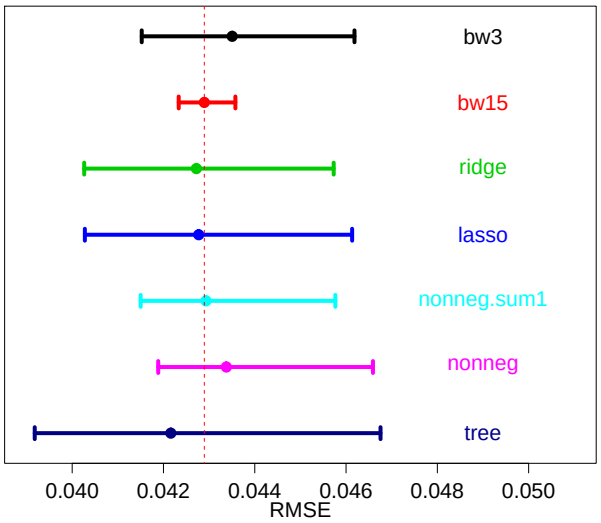


Figure 16: Comparing the middle 95% range of RMSE of all techniques. The dots represent the means. The red dotted line is the mean of *bw15*, which is used as benchmark

The next two pages show plots of fitted values for the whole range of $X^{(0)}$ using ridge regressions (p. 36) and regression trees (p. 37).

Regarding the fitted values using ridge regressions, while the coefficients in Table 8 make it hard to draw any conclusion, they start to tell a story when we plot the fitted values generated with those exact coefficients in Figures 17 and 18. Indeed, Figure 18 shows more clearly that the final fitted values tend to follow some sort of "balance" between local and longer-term trends: the curve is more sensible to local variations in the data than *bw15* and yet not as noisy as *bw3*.

Regarding the fitted values using regression trees, we know by looking at the trees on Figure 14 that most lines would only depend on the fitted values of *bw3*. Since those trees only have two terminal nodes, the final curve will be much "simpler" (there are only two possible values). Indeed this is what we see on Figures 19 and 20: most of the curves are flat for approximately $X^{(0)} \in [0; 55]$ and then bounce back and forth between $Y \approx 0.37$ and $Y \approx 0.44$ for the rest of $X^{(0)}$.

Finally, Figure 21 and 22 show fitted values of *bw3*, *bw15*, *ridge* and *tree* together on the same plot. When the problem at hand was introduced in Section 3.4, we wanted to find a smoother that would "tell a story" out of the noisy data. Now is the time to look at those stories.

The curve *bw15* shows a small (and maybe insignificant) decline of the support for Trump for the first month or so of 2016, followed by a steady and uninterrupted increase until the end of March where it seems like the support starts to plateau. While this curve is very smooth and easy to interpret, it may fail to represent smaller variations along the way. In other words, the increase in support for Trump from the beginning of February until the end of March 2016 wasn't necessarily perfectly continuous and linear. Indeed, it appears that the *ridge* curve does a better job of showing the ups and downs of Trump while also clearly demonstrating the perceptible upwards longer-trend. One could potentially match the local variations with real-life events (debates, states primaries results, speeches, etc.), but that is for another study.

When it comes to the *tree* curve, even if on average this learning scheme returns significantly lower errors, it is difficult to defend a scenario where the support for Trump stays relatively the same for about a month and a half, then suddenly jumps about 7% on a single day only to go back to its initial level a few days later, and jump back again shortly after. This would potentially mean that some specific events dramatically swung voting intentions while the support for Trump remained stable between those events. It seems fair to assume that the *ridge* curve gives a better picture of the evolution of voting intentions.

In summary, from a pure aesthetics and interpretability point of view, *ridge* seems to be the best compromise between short and long-term trends, even though *tree* is technically closer to the "truth" (i.e. has a lower mean RMSE).

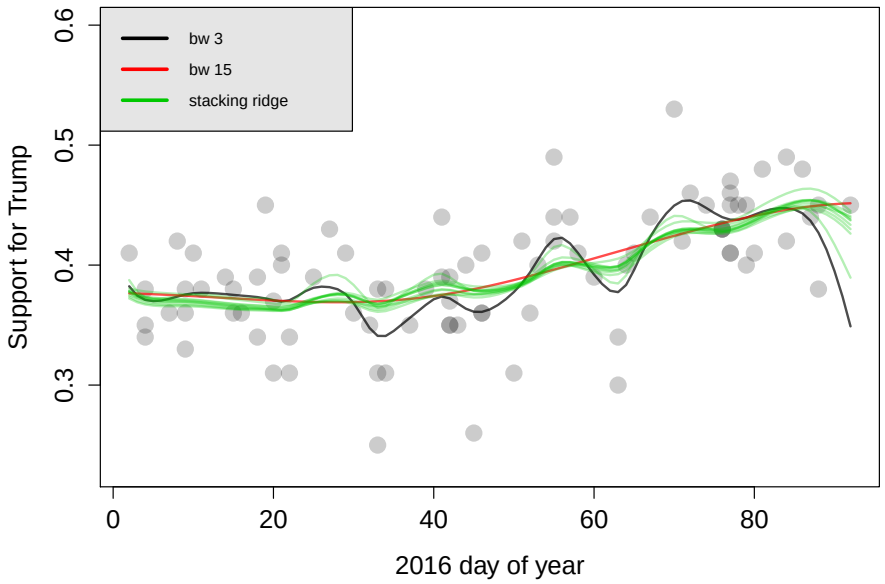


Figure 17: 10 lines of fitted values for the whole range of data when using ridge regression as level-1 generalizer. The coefficients used are in Table 8. *bw3* and *bw15* included as reference

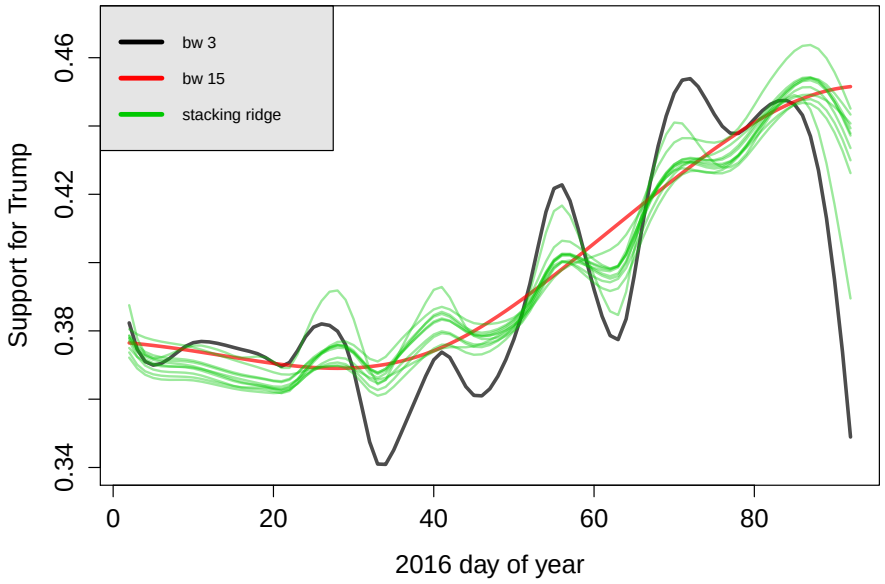


Figure 18: Close-up of Figure 17

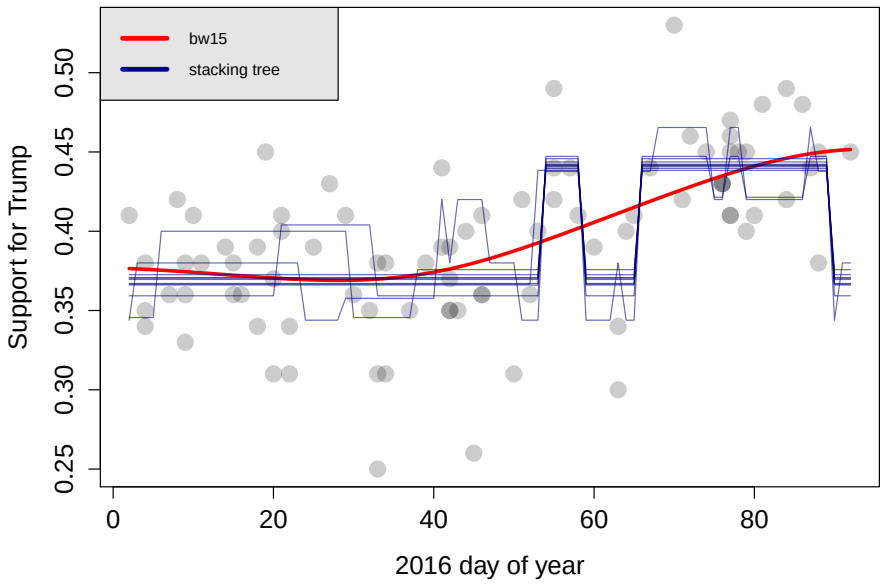


Figure 19: Fitted values for the whole range of data when using regression trees as level-1 generalizer. Each line corresponds to a fold in Figure 14. *bw15* is plotted as reference.

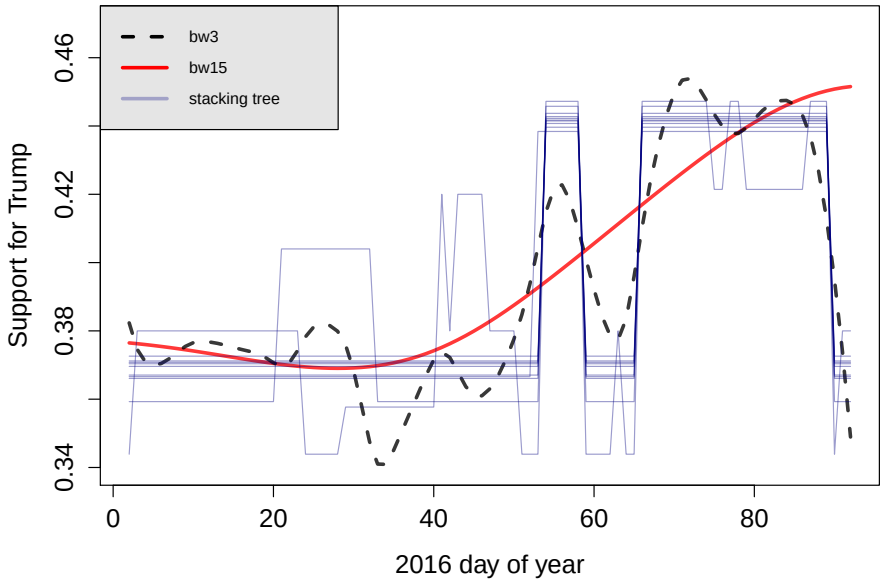


Figure 20: Close-up of Figure 19. *bw3* and *bw15* are plotted as reference.

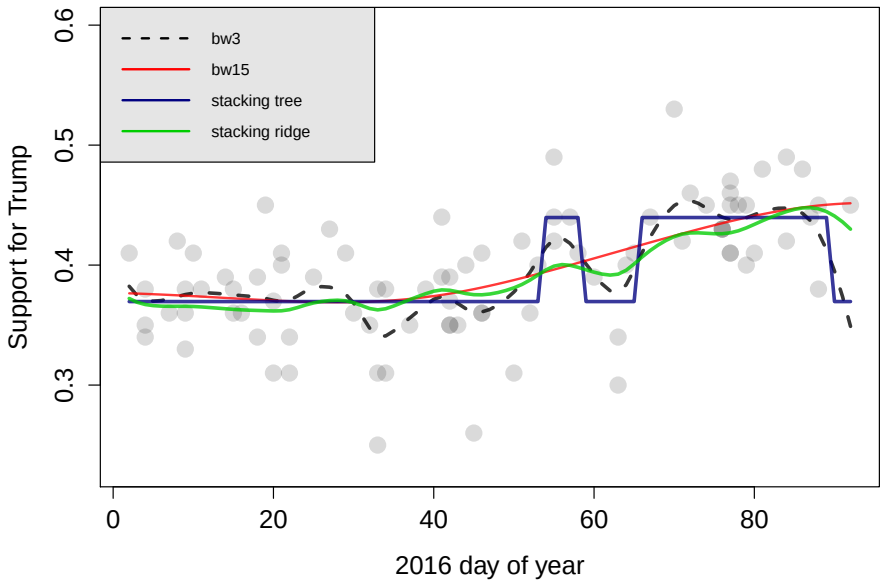


Figure 21: Fitted values of *bw3*, *bw15*, *ridge* and *tree* over the domain of $X^{(0)}$. The *ridge* and *tree* curves are one of the ten curves of Figures 17 and 19 respectively which seemed to appropriately represent the general behaviour of their respective techniques.

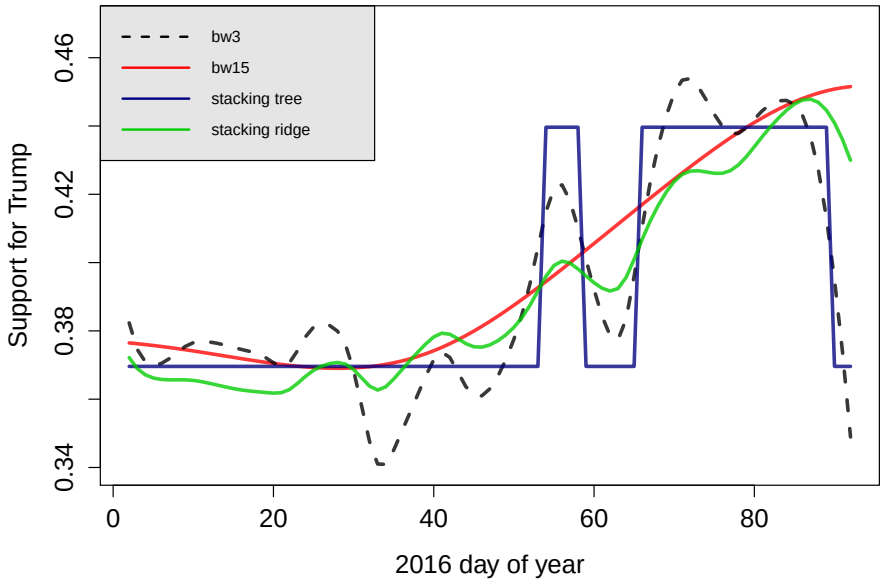


Figure 22: Close-up of Figure 21

4 Conclusion

This paper began by introducing stacked generalization through a literature review. The concept itself, along with an accompanying notation, were presented. Finally, a discussion about when stacked generalization works, the known issues and some variations of the original technique presented by Wolpert (1992) followed.

Following that first section, we suggested that stacked generalization could be employed for a task that it is not usually used for: parameter selection. Indeed, the hypothesis was as follows: when choosing a parameter for a learning scheme in a supervised-learning context, if a cross-validation returns unstable results for the choice of that parameter, then it can be beneficial to aggregate the fitted values of the learning scheme using different parameters rather than simply selecting the best parameter. We suggested that stacked generalization could be use for such aggregation. In other words, one could use the different sets of fitted values as inputs in another learning scheme and learn at the meta level (level-1).

Since significant knowledge in the field of stacked generalization is derived from experiments, we decided to conduct our own experiment in order to test the above hypothesis. Our results confirm that in a situation where parameter selection is unstable, it is possible to obtain lower errors by learning at the meta level. Generally speaking, in the context of our problem, the final fitted values turned out to be a balance between low and high bandwidths, which is consistent with Breiman's (1996b) finding that stacking is most efficient when dissimilar sets of fitted values are used as inputs at level-1. Different learning schemes, linear and non-linear ones, were used at the meta level. We found that for our problem, on average, regression trees return the lowest errors while ridge regressions return the most interpretable final fitted values. It is unclear if this applies only to this problem or if this conclusion can be generalized to other problems.

Finally, the goal of the experimentation part was mostly to confirm whether it is *possible* to obtain lower errors with stacking in the context presented in the hypothesis (unstable parameter selection). The experiment was able to confirm this, hence leaving the door open for further research. As a final word, we present a few of those research opportunities.

First, the experiment could be replicated on different datasets. It would be interesting to see if ridge regressions also tend to work better than constrained linear regressions on other datasets, or if this was specific to the 2016 Republican Primary Surveys data. One could also try to apply some constraints on the coefficients of a ridge regression. Second, other non-linear methods could be used at level-1. For a smoothing problem, one could try to find a model which returns smoother fitted values than regression trees and which also learns which sets of fitted values is most precise on which region of the domain of $X^{(1)}$. Third, regarding the multi-collinearity problem, one could try to use dimensionality reduction techniques (e.g. principal component analysis) before fitting $G^{(1)}$ and see if this works better than the other solutions used so far (ridge, lasso and constraints on the coefficients). Fourth, the explanatory variables at level-0 could be used at level-1 as well. This has been done in Chan and Stolfo (1995), notably, but the authors' technique, *attribute-combiner*, is slightly different than stacking. Lastly, there is a multitude of learning schemes which require the selection of a parameter. One could perform similar experiments for other parameters and try to draw more general conclusions about using stacked generalization for parameter selection.

References

Armstrong, J. S. (2001). Combining forecasts. In *Principles of forecasting*, pages 417–439. Springer.

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

Breiman, L. (1996b). Stacked regressions. *Machine learning*, 24(1):49–64.

Breiman, L. et al. (1998). Arcing classifier (with discussion and a rejoinder by the author). *The annals of statistics*, 26(3):801–849.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Chan, P. K. and Stolfo, S. J. (1995). A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the 12th International Conference on Machine Learning*, pages 90–98. Morgan Kaufmann.

Cronbach, L. J. and Meehl, P. E. (1955). Construct validity in psychological tests. *Psychological bulletin*, 52(4):281.

Džeroski, S. and Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

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

Hu, M. Y. and Tsoukalas, C. (2003). Explaining consumer choice through neural networks: The stacked generalization approach. *European Journal of Operational Research*, 146(3):650–660.

HuffingtonPost (2016). Poll Chart 2016 National Republican Primary. `http://elections.huffingtonpost.com/pollster/2016-national-gop-primary` [Accessed 4 April 2016].

Jahrer, M., Töscher, A., and Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 693–702. ACM.

Lawson, C. and Hanson, R. (1995). Solving least squares problem. *Classics in applied mathematics (SIAM, Philadelphia, PA)*.

LeBlanc, M. and Tibshirani, R. (1996). Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91(436):1641–1650.

Ness, S. R., Theocharis, A., Tzanetakis, G., and Martins, L. G. (2009). Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 705–708. ACM.

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

Smyth, P. and Wolpert, D. (1997). Stacked density estimation. In *NIPS*, pages 668–674. Citeseer.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, pages 111–147.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Ting, K. M. and Witten, I. H. (1997). Stacking bagged and dagged models. In *14th International Conference on Machine Learning*, pages 367–375. Morgan Kaufmann.

Ting, K. M. and Witten, I. H. (1999). Issues in stacked generalization. *J. Artif. Intell. Res.(JAIR)*, 10:271–289.

Töscher, A., Jahrer, M., and Bell, R. M. (2009). The bigchaos solution to the netflix grand prize. `http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf` [Accessed June 2016].

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

Appendices

A Dataset: Republican primary surveys

ID	pollster	day.of.year	Trump
1	Morning Consult	92	0.45
2	NBC/SurveyMonkey	88	0.45
3	IBD/TIPP	88	0.38
4	Ipsos/Reuters	87	0.44
5	YouGov/Economist	86	0.48
6	NBC/SurveyMonkey	81	0.48
7	Pew	77	0.41
8	Morning Consult	84	0.49
9	PPP (D)	84	0.42
10	Ipsos/Reuters	79	0.45
11	McLaughlin (R)	77	0.45
12	FOX	80	0.41
13	Bloomberg/Selzer	79	0.4
14	Morning Consult	78	0.45
15	Quinnipiac	76	0.43
16	CBS/Times	77	0.46
17	CNN	77	0.47
18	Monmouth University	77	0.41
19	NBC/SurveyMonkey	74	0.45
20	Morning Consult	76	0.43
21	Rasmussen	76	0.43
22	Ipsos/Reuters	72	0.46
23	Morning Consult	71	0.42
24	NBC/SurveyMonkey	67	0.44
25	YouGov/Economist	70	0.53
26	Ipsos/Reuters	65	0.41
27	Morning Consult	64	0.4
28	ABC/Post	63	0.34
29	NBC/WSJ	63	0.3
30	NBC/SurveyMonkey	60	0.39
31	Ipsos/Reuters	58	0.41
32	NBC/SurveyMonkey	53	0.4
33	Morning Consult	57	0.44
34	CNN	55	0.49
35	YouGov/Economist	55	0.44
36	Morning Consult	55	0.42
37	Ipsos/Reuters	51	0.42
38	IBD/TIPP	50	0.31
39	Rasmussen	52	0.36
40	NBC/SurveyMonkey	46	0.36
41	FOX	46	0.36
42	Ipsos/Reuters	44	0.4
43	McLaughlin (R)	42	0.35
44	Morning Consult	46	0.41
45	NBC/WSJ	45	0.26
46	CBS	43	0.35
47	Robert Morris University	42	0.37
48	Suffolk/USA Today	42	0.35
49	YouGov/Economist	42	0.39
50	Quinnipiac	41	0.39
51	NBC/SurveyMonkey	39	0.38
52	Morning Consult	41	0.44
53	Ipsos/Reuters	37	0.35
54	Morning Consult	34	0.38
55	NBC/SurveyMonkey	32	0.35
56	Rasmussen	34	0.31
57	Quinnipiac	33	0.31
58	Morning Consult	33	0.38
59	PPP (D)	33	0.25
60	Ipsos/Reuters	30	0.36
61	Morning Consult	29	0.41
62	NBC/SurveyMonkey	25	0.39
63	YouGov/Economist	27	0.43
64	IBD/TIPP	22	0.31
65	Bloomberg/Purple Strategies	22	0.34
66	CNN	21	0.41
67	Morning Consult	21	0.4
68	ABC/Post	20	0.37
69	Public Religion Research Institute	20	0.31
70	NBC/SurveyMonkey	18	0.39
71	FOX	18	0.34
72	Zogby (Internet)	19	0.45
73	Ipsos/Reuters	16	0.36
74	YouGov/Economist	15	0.38
75	Monmouth University	15	0.36
76	Morning Consult	14	0.39
77	NBC/SurveyMonkey	11	0.38
78	Ipsos/Reuters	9	0.38
79	NBC/WSJ	9	0.33
80	YouGov/Economist	9	0.36
81	Gravis Marketing/One America News	10	0.41
82	Morning Consult	8	0.42
83	CBS/Times	7	0.36
84	NBC/SurveyMonkey	4	0.38
85	IBD/TIPP	4	0.34
86	FOX	4	0.35
87	Ipsos/Reuters	2	0.41

Table 13: Dataset collected by HuffingtonPost (2016). The column *Trump* represents the percentage of approval for Donald Trump during the 2016 Republican Primary race. The column *day.of.year* represents the day on which the poll was started (in 2016).

B R code

All R code, along with a CSV file containing the dataset in Appendix A, can be found on GitHub at **http://bit.ly/lse-dissertation**

C Statistical tests

C.1 RMSEs of bw15 versus ridge

F test to compare two variances

```
data:  RMSEs.bw15 and RMSEs.ridge
F = 0.054478, num df = 999, denom df = 999, p-value < 0.00000000000000022

alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.04812029 0.06167556

sample estimates:
ratio of variances
0.05447794
```

Table 14: F-test comparing the variances of the RMSEs generated with bandwidth 15 and with stacking using a ridge regression

Welch Two Sample t-test

```
data:  RMSEs.bw15 and RMSEs.ridge
t = 4.0143, df = 1107.5, p-value = 0.00006364

alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
0.00008913094 0.00025956816

sample estimates:
mean of x  mean of y
0.04289494 0.04272059
```

Table 15: Welch Two Sample t-test comparing the means of the RMSEs generated with bandwidth 15 and with stacking using a ridge regression

C.2 RMSEs of ridge versus lasso

F test to compare two variances

```
data:  RMSEs.ridge and RMSEs.lasso
F = 0.91735, num df = 999, denom df = 999, p-value = 0.173

alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.8102947 1.0385509

sample estimates:
ratio of variances
0.9173507
```

Table 16: F-test comparing the variances of the RMSEs generated with stacking using ridge and lasso regressions

Two Sample t-test

```
data:  RMSEs.ridge and RMSEs.lasso
t = -0.83793, df = 1998, p-value = 0.4022

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
-0.00017115602  0.00006868146

sample estimates:
mean of x  mean of y
0.04272059 0.04277183
```

Table 17: Two Sample t-test comparing the means of the RMSEs generated with stacking using ridge and lasso regressions

C.3 RMSEs of bw15 versus non.neg.sum.1

F test to compare two variances

```
data:  RMSEs.bw15 and RMSEs.nonneg.sum1
F = 0.089825, num df = 999, denom df = 999, p-value < 0.00000000000000022

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:
0.07934198 0.10169225

sample estimates:
ratio of variances
0.08982463
```

Table 18: F-test comparing the variances of the RMSEs generated with *bw15* and with regularized linear stacking (non-negativity and sum-to-one constraints)

Welch Two Sample t-test

```
data:  RMSEs.bw15 and RMSEs.nonneg.sum1
t = -1.8819, df = 1177, p-value = 0.0601

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
-0.00013217547  0.00000275454

sample estimates:
mean of x  mean of y
0.04289494 0.04295965
```

Table 19: Welch Two Sample t-test comparing the means of the RMSEs generated with *bw15* and with regularized linear stacking (non-negativity and sum-to-one constraints)

C.4 RMSEs of non.neg.sum.1 versus non.neg

F test to compare two variances

```
data:  RMSEs.nonneg.sum1 and RMSEs.nonneg
F = 0.91689, num df = 999, denom df = 999, p-value = 0.1705

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:
0.8098914 1.0380340

sample estimates:
ratio of variances
0.9168941
```

Table 20: F-test comparing the variances of the RMSEs generated with *nonneg.sum1* and *non.neg*

Two Sample t-test

```
data:  RMSEs.nonneg.sum1 and RMSEs.nonneg
t = -9.7726, df = 1998, p-value < 0.00000000000000022

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
-0.0005332110 -0.0003549718

sample estimates:
mean of x  mean of y
0.04293304 0.04337713
```

Table 21: Two Sample t-test comparing the means of the RMSEs generated with *nonneg.sum1* and *nonneg*

C.5 RMSEs of bw15 versus tree

F test to compare two variances

```
data:  RMSEs.bw15 and RMSEs.tree
F = 0.025688, num df = 999, denom df = 999, p-value < 2.2e-16

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:
0.02269058 0.02908241

sample estimates:
ratio of variances
0.02568846
```

Table 22: F-test comparing the variances of the RMSEs generated with *bw15* and *tree*

Welch Two Sample t-test

```
data:  RMSEs.bw15 and RMSEs.tree
t = 11.764, df = 1050.3, p-value < 2.2e-16

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
0.0006114254 0.0008562306

sample estimates:
mean of x  mean of y
0.04289494 0.04216111
```

Table 23: Welch Two Sample t-test comparing the means of the RMSEs generated with *bw15* and *tree*

C.6 RMSEs of ridge versus tree

F test to compare two variances

```
data:  RMSEs.ridge and RMSEs.tree
F = 0.47154, num df = 999, denom df = 999, p-value < 2.2e-16

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:
0.4165095 0.5338383

sample estimates:
ratio of variances
0.4715387
```

Table 24: F-test comparing the variances of the RMSEs generated with *ridge* and *tree*

Welch Two Sample t-test

```
data:  RMSEs.ridge and RMSEs.tree
t = 7.488, df = 1769.8, p-value = 1.098e-13

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
0.0004129354 0.0007060214

sample estimates:
mean of x  mean of y
0.04272059 0.04216111
```

Table 25: Welch Two Sample t-test comparing the means of the RMSEs generated with *ridge* and *tree*

D Lambdas (λ) for ridge and lasso regressions

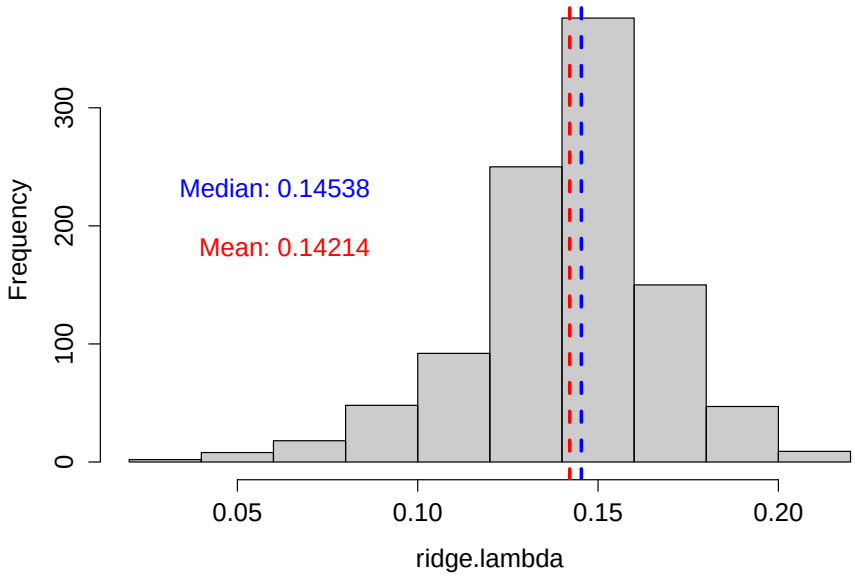


Figure 23: Plot of the frequencies of λ for the 1000 replications of Algorithm 3 when using a **ridge regression** at level-1

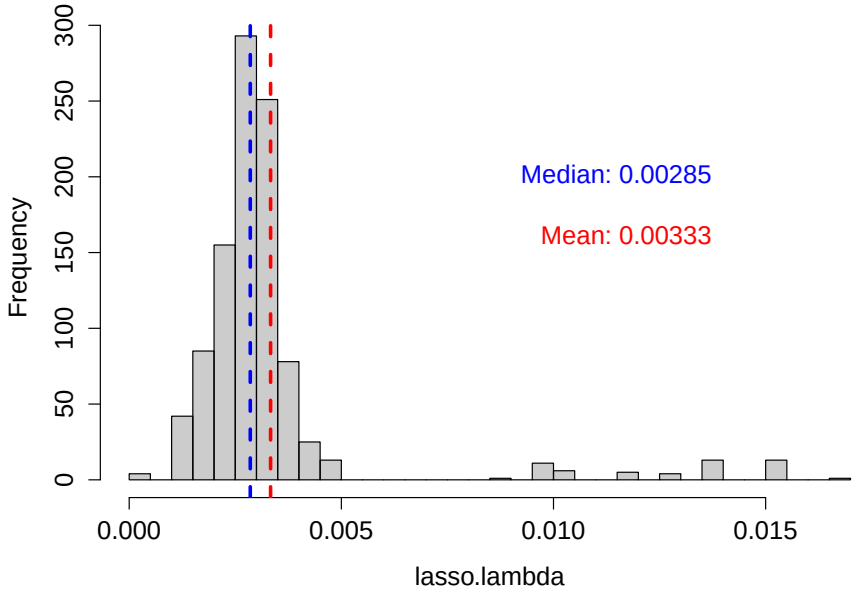


Figure 24: Plot of the frequencies of λ for the 1000 replications of Algorithm 3 when using a **lasso regression** at level-1