Detecting electoral fraud using machine-learning algorithms*

Lion Behrens†

August 15, 2019

Abstract. This research plan outlines an agenda to advance the use of machine-learning methods for the detection of election fraud. While former attempts have merely engaged in binary classification separating "clean" from "tainted" voting returns, this information alone does not tell election scholars much about which manipulation tactics have been in place and how intensively these might have shaped vote outcomes. Hence, I seek to develop classifiers that, first, can distinguish between four different *mechanisms of fraud*, namely the manual alteration of vote counts constructing new figures, the rotation of real figures from one party to another, the invalidation of votes and stuffing the ballot box. Second, these models should additionally be able to quantify the *extent* of fraudulent interference. Employed methods include penalized regression models, the naive Bayes classifier, random forests and feed-forward neural networks. Classifiers are trained using synthetic data that mimic empirical voting returns under different degrees of fraud. Their performance is evaluated over a large number of scenarios using Monte Carlo simulation techniques.

Keywords: *Election forensics, fraud detection, Machine learning, Monte Carlo simulation.*

*Manuscript prepared to document the project's current state for the *Deutscher Akademischer Austauschdienst.*

†Collaborative Research Center 884 "The Political Economy of Reforms", Graduate School of Economic and Social Sciences (GESS), behrens@uni-mannheim.de.

1 Introduction

Can we successfully exploit algorithms developed in the machine-learning literature to the detection of election fraud? Since the beginning of the early 2000s, an ever-increasing body of literaure seeks to develop statistical methods that are designed to flag peciuliarities in voting returns which are indiciative of systematic interference. Some of the proposed approaches are rather simple focusing on the distribution of digits (**?**, Beber and Scacco 2012) or turnout and vote shares (Myagkov et al. 2009). Other scholars have developed more sophisticated methods that exploit Monte Carlo simulation to quantify the degree of interference that is thought to be inherent in manipulated voting returns (Klimek et al. 2012, Mebane and Klaver 2015).

Despite their promising applications in modeling complex data structures across a wide range of fields, remarkably little effort has so far been placed in exploiting machine-learning techniques for the detection of electoral fraud. Evidently, this should catch forensic scholars by surprise. First, this neglection seems surprising since the literature on election forensics has so far advanced a somewhat bewildering landscape of methods to detect election anomalies. Throughout these contributions, little guidance exists on the scenarios in which each of the developed tools should be applied. Hence, it is unclear in which situations it is actually best to apply digit tests (such as those inspired by Benford's law) and when these outperform distributional methods that focus on turnout and vote share. Second, an increasing number of authors has doubted the power of conventional forensics tests for the data structures that election researchers usually have at hand (**?**) or have expressed worries that these methods mit yield a vast share of false-positive results (**?**, Rozenas 2017). Against this background, sophisticated machine-learning techniques can serve as a *unifying framework* for many of the developed approaches, as these can easily directly incorporate their logic and indices into their learners. By borrowing algorithms from the machine-learning literature, we can set up methods that detect anomalies in election returns combining a wide range of individual methods.

From surveying the literature, only four contributions have started to exploit machine-learning techniques for the detection of electoral fraud. Somewhat related to the field of election forensics, Cantú (forthcoming) exploits an original image database of the vote-tally sheets from Mexico's 1988 presidential election and uses Convolutional Neural Networks (CNN) to analyze all collected pictures. Given that the vote-counting process was in the hands of district officials, he takes manual alterations of any hand-written number as a symbol for irregularities and uses his classifier in order to graphically identify those vote tallies where alterations were present. While this constitutes a laudable effort, any application of such a procedure hinges on the availability of original vote-tally sheets (or scans thereof) across all electoral units. Evidently, researchers are typically not in possession of such data. A more promising approach lies in the direct application of machine-learning techniques to official voting returns as made publicly available after a given election took place. First, Montgomery et al. (2015) have combined information on digit distributions with country-specific context-variables to detect election irregularities in a large cross-national dataset. Second, two studies have used synthetic data to train fraud detection algorithms where the amount of interference is specified by the researcher and known. Cantú and Saiegh (2011) employ a naive Bayes classifier as their learning algorithm and externally evaluate it using a data set of district-level vote counts in the province of Buenos Aires (Argentina) between 1931 and 1941, for which detailed documentations of fraud are available. They

conclude that the elections considered to be irregular (legitimate) by most historical accounts are unambiguously classified as fraudulent (clean) by the learner. Moreover, exploiting artificial data and returns on Argentina's primary election for national deputies in 2013, Levin et al. (2016) examine the power of a variety of classification approaches such as the K-Means-Classifier, K-Nearest Neighbor Algorithm, and tree-based approaches like random forests.

Hence, while first attempts do exist in using machine-learning algorithms for the field of election forensics, this endeavor is still in its *very* first steps. The second part of this dissertation seeks to push forward the applicability of machine-learning techniques for the analysis of standalone voting returns. To this end, this section sketches the research questions that I seek to tackle within this individual project (Section 2.2), provides a brief overview potential methodological strategies (Sections 2.3) and sketches a study design that can be made use of for such an endeavor (Section 2.4).

2 Research Questions

Only very initial attempts have started to exploit machine-learning techniques for the identification of data anomalies that are indicative or manual manipulation in voting returns. Hence, research in this area *could* focus on a variety of smaller questions that are underlying future endeavors in this field. For instance, as researchers are often confronted with voting returns that consist of fewer electoral units than many of the developed statistical approaches are in need of to yield sufficient power, it is an open question which sample sizes of electoral units one is actually in need of when applying machine-learning techniques to the field of election forensics. Second, often published results are presenting aggregated data (for instance results for electoral precincts) that do not document each variable of interest on the most basic unit of analysis (say polling stations) where interference is likely to take place but on higher levels. Aggregated data hence most likely consist of a mix of clean and tainted underlying units within one *observed* unit of analysis. This might easily blurr the anomalies that are inherent in published returns. Research could therefore focus on the levels of aggregation that machine-learning techniques are in need of and whether these can be trained to detect election irregularities on aggregated at all.

However, while such questions might easily turn into the focus of interest of this project at some point, I predominantly aim for two main contributions that would vastly put forward the application of machine-learning algorithms for election forensics. First, importantly, current machine-learning applications have trained learners predominantly for binary classification of election results, aimed at separating voting returns that are flagged as "tainted" from those that are predicted to be "clean" (Cantú and Saiegh 2011, Montgomery et al. 2015). Evidently, this comes with the major shortcoming that a clear elaboration on *why* such are classified the way they are and an indication of the underlying *mechanism* of interference is missing from this binary assessment. Against this background, I extend current approaches to *categorical* classification not only flagging specific voting returns as tainted, but classifying them regarding four specific mechanisms of fraud: Manual alteration of vote counts constructing new figures, the rotation of real figures from one party to another, the invalidation of votes and stuffing the ballot box. Second, I extend the current approach to binary classification of voting returns to systematically predicting the *extend* of contamination. That is, my learners should

not only classify the concrete mechanism of manipulation, but also allow a probabilistic statement about which percentage of the total vote count is affected.

3 Synthetic data generation and measurements

This section sketches how algorithms developed in the literature on machine-learning can actually be used for the detection of different manipulation strategies by training them on synthetic data where the "true outcomes" (the mechanism of interference as will as its extent) are known. First, I elaborate on the data generating methodology that is underlying the synthetic creation of clean and tainted voting returns (Section 2.3.1). Second, I sketch how fraud can be measured by concrete dependent variables and what the feature space that those are mapped to might look like (Section 2.3.2). Third, I present an exemplary methodical approach for the detection of electoral interference, Bayesian additive regression trees (Section 2.3.3).

3.1 Data generating methodology

In order to train machine-learning algorithms to automatically detect the (i) type and (ii) extent of fraudulent interference in voting returns, we are in need of data for which the (i) existence, (ii) type and (iii) extent of manual manipulation is known. Although there are several historical cases where election malpractice is well-documented (**?**), these documents are far from constituting a valid data source to train any kind of classifier. Therefore, I train my classifiers using synthetic data on a large number of electoral contests that are generated from a series of Monte Carlo simulations (similar to initial approaches in Cantú and Saiegh 2011 and Levin et al. 2016). The data generating process for these synthetic data structures is defined by two steps. First, baseline data structures are constructed that are sought to resemble a variety of features that are known to be inherent in clean voting returns (Beber and Scacco 2012, Klimek et al. 2012). Second, taking a given share of electoral units p, manipulation to these synthetic baseline data sets is executed manually.

GENERATING CLEAN VOTING RETURNS

The two basic features that my synthetic and clean voting returns should satisfy is that (i) their vote counts follow Benford's law and that (ii) turnout and party vote shares are characterized by two orthogonal normal distributions. Formally, Benford's law expresses that the probability of the digit d ($d \in 0, 1, 2, .., 9$) arising in the nth position ($n > 1$) is

$$\sum_{k=10^{n-2}}^{10^{n-1}} log_{10}(1 + \frac{1}{10k + d}). \tag{1}$$

For processes that are thought to follow Benford's law, Table 1 documents the predicted frequencies for first and second digits. Moreover, across all electoral units, turnout and vote shares are argued to be characterized by two orthogonal normal distributions. Hence, the clean baseline data structures are generated in two steps. First, $m = 10,000$ sets of voting returns (each representing on election) are

Table 1: Predicted frequencies by Benford's law.

	0	1	2	3	4	5	6	7	8	9
First Digit	-	0.301	0.176	0.125	0.097	0.079	0.067	0.058	0.051	0.046
Second Digit	0.120	0.114	0.109	0.104	0.100	0.097	0.093	0.090	0.088	0.085

constructed, where each generated dataset consists of $n = 200, ..., k$ individual electoral units. In a first scenario, two parties j ($j \in A, B$) are competing in each unit. To satisfy the first condition, following Cantú and Saiegh (2011), the total number of votes for party j in the electoral unit $i = \{1, ..., n\}$ is described by:

$$V_{ji} = a_j X_j, \tag{2}$$

where a_j is a constant and X_j is a random variable following Benford's distribution. Notably, the total number of votes V_{ji} consists of a systematic and a stochastic component. The first component a_j determines the general support for party j across all electoral units. If party A receives more electoral support than party B, then $a_A > a_B$. Multiplying the second component X_j with the constant a_j creates the actual vote counts for each party. Through this procedure, different electoral results are determined through the specification of a_j while the random variable X_j ensures that the clean voting returns indeed follow digit distributions as specified by Benford's law.

To satisfy the second condition, we need to ensure that turnout and party specific vote shares follow two normal distributions and that these are orthogonal to each other. Therefore, the concrete value of a_j in Equation 2 is drawn from a Gaussian process defined by

$$a_j \sim N(a_j, \sigma^2_{a_j}). \tag{3}$$

This ensures that, additionally to the concrete vote counts following Benford's distribution, vote shares are normally distributed. Finally, I incorporate the abstinence of a certain share of voters from the electoral event and model turnout. Therefore, from each electoral unit i, all figures are multiplied with turnout t_j where

$$t_i \sim N(t_i, \sigma^2_{t_i}) \tag{4}$$

and $0 < t_i < 1$. Next to introducing normally distributed turnout figures, the separate generation of vote shares and turnout rates by two independent Gaussian processes ensures that the distribution of party vote shares and turnout is kept orthogonal.

Table 2: Four different manipulation strategies on twelve exemplary electoral units.

	Original			Counts Altered			Counts Switched			Ballots Invalidated			Ballots Added		
Unit ID	Party A	Party B	N	Party A	Party B	N	Party A	Party B	N	Party A	Party B	N	Party A	Party B	N
1	152	211	363	152	211	363	152	211	363	152	211	363	152	362	514
2	918	396	1314	718	596	1314	396	918	1314	318	396	714	918	722	1640
3	712	520	1232	712	520	1232	712	520	1232	712	520	1232	712	520	1232
4	1132	235	1367	1132	235	1367	1132	235	1367	1132	235	1367	1132	235	1367
5	250	619	869	250	619	869	250	619	869	250	619	869	250	619	869
6	287	540	827	107	720	827	287	540	827	157	540	697	540	832	1372
7	398	217	615	198	417	615	217	398	615	118	217	335	398	625	1023
8	210	699	909	210	699	909	210	699	909	210	699	909	210	699	909
9	113	544	657	113	544	657	113	544	657	113	544	657	113	544	657
10	685	574	1259	685	574	1259	685	574	1259	685	574	1259	685	574	1259
11	771	362	1133	521	612	1133	362	771	1133	221	362	583	771	763	1534
12	1665	855	2520	1665	855	2520	1665	855	2520	1665	855	2520	1665	855	2520
N	7293	5772	13065	6463	6602	13065	6181	6884	13065	5733	5772	11505	7546	7350	14896

Note: Four different manipulation strategies favoring Party B over Party A. Manipulated units shaded in grey. Alteration and rotation of vote counts leave turnout fixed. Ballot invalidation and addition affect turnout as well.

MANUALLY MANIPULATING VOTING RETURNS

As a second step, a given proportion of p clean baseline data structures are artificially tainted. Since I am introducing four separate manipulation mechanisms, each mechanism is applied to a proportion of $p/4$ of the $m = 10,000$ returns. To incorporate vote switching, the generated vote counts for both parties are simply reversed in a share of λ units. To mirror the alteration of vote counts (while leaving turnout stable), a fixed proportion of $\lambda > 0$ of party j's votes is taken away and added to the other party's vote share. To incorporate the invalidation of ballots, a proportion of λ votes of party j is invalidated and thus subtracted from this party's vote count. Lastly, ballot box stuffing is mirrored by adding a fixed proportion of λ additional votes to party j's vote count. An exemplary representation of these four mechanism is portrayed in Table 2.

3.2 Measurements: Fraud and Feature Space

MEASURING FRAUD

The procedure that was described in the foregoing section ends up in $m = 10,000$ sets of voting returns out of which a proportion p was manipulated and a proportion $1 - p$ voting returns is left untouched. Out of the proportion of p manipulated sets, each type of fraud mechanism is balanced out and observed in 25% of the tainted sets. Within each tainted set, the extent of manipulation is defined by λ, with interference being introduced in a share of λ electoral units. Given that artificial data has been constructed, two separate learners are trained on two respective dependent variables. To identify the mechanism of manipulation, my dependent variable consists of a nx1 column vector where $y^t \in \{\text{Clean, Counts Altered, Counts Switched, Ballots Invalidated, Ballots Added}\}$ and n denotes the number of analyzed voting returns. To estimate λ, that is, the severity of interference, y^e denotes the percentage of votes that were altered, added, or deleted by each mechanism, with $y^e = 0$ if a dataset was left untouched.

$$y^t = \begin{bmatrix} y_1^t = & \text{clean} \\ y_2^t = & \text{counts altered} \\ y_3^t = & \text{clean} \\ y_5^t = & \text{clean} \\ y_6^t = & \text{ballots added} \\ y_7^t = & \text{clean} \\ y_8^t = & \text{ballots invalidated} \\ y_9^t = & \text{clean} \\ y_{10}^t = & \text{clean} \\ y_{11}^t = & \text{clean} \\ y_{12}^t = & \text{counts switched} \\ \dots & \dots \\ y_m^t = & y_m^t \end{bmatrix} \qquad y^e = \begin{bmatrix} y_1^e = & 0 \\ y_2^e = & 0.08 \\ y_3^e = & 0 \\ y_5^e = & 0 \\ y_6^e = & 0.24 \\ y_7^e = & 0 \\ y_8^e = & 0.16 \\ y_9^e = & 0 \\ y_{10}^e = & 0 \\ y_{11}^e = & 0 \\ y_{12}^e = & 0.03 \\ \dots & \dots \\ y_m^e = & y_m^e \end{bmatrix}$$

INDEPENDENT VARIABLES

The promising feature of combining techniques developed in the machine-learning literature with election forensics lies that we might exploit the power of these algorithms solely by feeding them with information that is inherent in the voting data at hand. Hence, the independent variables that are used to identify the mechanism of interference and to estimate the degree of contamination are a collection of summary statistics describing the distribution of individual variables and their interrelation. Table 3 portrays a summary of the potential feature space that might be used. From this table it also becomes clear how machine-learning can serve as a unifying framework to the current field of election forensics, as features can incorporate different kinds of characteristics that are inherent to voting returns which individual forensic methods have been built around.

3.3 Methodical Kick-Off: Bayesian Additive Regression Trees

How can the presented feature space be exploited for the detection of electoral interference? When training learners that should (i) identify the mechanism with which clean voting returns have been tainted and (ii) estimate the degree of interference that was introduced, I am dealing with both regression and classification tasks. Hence, in order to exploit a given feature space that is mapped to y, I am in need of methodical approaches that can satisfy both tasks. Within the machine-learning literature, a vast array of methods has been developed for these purposes. This project will certainly explore a set of diverse methodological approaches. One primising venue is provided by Bayesian additive regression trees (BART, Chipman et al. 2012) concern the fundamental problem to estimate an unknown function f to predict an output Y using a d dimensional vector of inputs $\boldsymbol{X} = (x_1, ..., x_d)$ where

$$Y = f(x) + \epsilon, \qquad \epsilon \; N(0, \sigma^2). \tag{5}$$

In the case presented here, the output is one of the two dependent variables y^t and y^e presented in Section 2.3.2 and the matrix of inputs consists of all variables that are presendet in Table 3. In the machine-learning literature, a wide range of approaches has been proposed to flexibly approximate this problem. For this set-up, tree-based models are suited exceptionally well because they avoid limitations to predictive power such as the fixed and inflexible functional form inherent to generalized linear models or the assumption of conditional independence among all inputs that underly approaches such as the naive Bayes estimator (Levin et al. 2016), while at the same time not requiring the enormous amounts of training data needed to fine-tune parameters such as those of neural networks with multiple hidden layers.

In essence, tree-based approaches approximate $f(x)$ by recursively partitioning the training data into k separate sets and then computing expected values $\boldsymbol{M} = (\mu_1, ..., \mu_k)$ for each separated unit of the sample space. These k partitions are accomplished by a series of inequality-constrained binary splits of the form $\{x_d > c\}$ and $\{x_d \leq c\}$ where c is a threshold estimated by the model (see Figure 3). By letting these splits minimize a loss function, this recursive splitting is achieved such that at

Table 3: Summary of the potential feature space.

		Mean	Chi	P05s	SD	Skewness	Curtosis	Correlation
Overall								
	Univariate							
	First Digit	X	X	X	X	X	X	
	Second Digit	X	X	X	X	X	X	
	Last Digit	X	X	X	X	X	X	
	Turnout	X			X	X	X	
	Invalid Votes	X			X	X	X	
	Bivariate							
	First Digit/Second Digit							X
	First Digit/Third Digit							X
	Second Digit/Third Digit							X
Party-Specific								
	Univariate							
	First Digit	X	X	X	X	X	X	
	Second Digit	X	X	X	X	X	X	
	Last Digit	X	X	X	X	X	X	
	Vote Share	X			X	X	X	
	Bivariate							
	First Digit/Second Digit							X
	First Digit/Third Digit							X
	Second Digit/Third Digit							X
	Turnout/Vote Share							X
	Invalid Votes/Vote Share							X

Note: Each "X" denotes one variable used for classification and prediction. "Overall" values refer to distributions over all electoral units and parties. "Party-Specific" values refer to distributions over all electoral units but are calculated for each party separately. "P05s" refers to the percentage of "0" and "5" figures within all first/second/third digits.

every stage of the process, the binary split results in two sets that are as homogeneous as possible, and begins with the indicator that exhibits most predictive power. The resulting process can be thought of as the evolution of a binary tree T from its root to all its terminal nodes, each of which is assigned with a prediction for Y based on its expected value μ_k in the respective set. This describes a non-parametric approach to flexibly approximate the non-linear function $f(x)$ allowing for high-level function complexities and covariate interactions. Figure 3 marks an example decision tree which partitions a sample space defined by three covariates (x_1, x_2, x_3) into six mutually exclusive sets and for each set outputs a prediction μ_k.

Given one particular observation, we can thus arrive at a prediction for Y by essentially 'moving down' the tree following all splitting rules with respect to its covariates until a terminal node is reached. The construction of one specific tree T can be captured by a function

$$Y = g(\boldsymbol{X}, T, \boldsymbol{M}) + \epsilon, \qquad \epsilon \ N(0, \sigma^2), \tag{6}$$

where T and $\boldsymbol{M}$ are the parameters to be estimated. Naturally, relying on a single tree when analyzing training data will vastly overfit the analyzed sample and eventually lead to low out-of-sample performances. Various (so called 'ensemble') methods amend this basic tree-model by using the sum of a *sequence of trees*

$$Y = \sum_{j=1}^{m} g(\boldsymbol{X}, T, \boldsymbol{M}) + \epsilon, \qquad \epsilon \ N(0, \sigma^2), \tag{7}$$

for prediction and have attracted much attention in recent years. Most prominent approaches include bagging and random forests, which both use randomization techniques to create a large number of independent trees whose predictions are subsequently averaged and long time constituted *the* state-of-the-art for problems as depicted by Equation 1. Recently, Chipman et al. (2012) proposed to deal with overfitting by introducing the BART model. This algorithm redefines current approaches in two notable ways. First, the BART algorithm is defined as a statistical model which inherits a prior and a likelihood function, letting Equation 3 become

$$Y \sim D(\sum_{j=1}^{m} g(\boldsymbol{X}, T, \boldsymbol{M})), \tag{8}$$

where D is some probability distribution to be chosen. Second, BART places highly influential regularization priors over the models parameters to be estimated. These amendments allow for full posterior inference and can produce measures of uncertainty regarding the model's parameters and additional quantities of interest. Additionally, rather than relying on randomization techniques, the priors limit both the size as well as the complexity of trees and thus efficiently deal with overfitting.

For the task considered here, BART are promising for two reasons. First, this framework is both suited for the regression task to determine the *degree* of contamination inherent to voting returns as well as for the categorical classification task to estimate the mechanism of interference as introduced in the synthetic data generation. Second, as Chipman et al. (2012) show, BART outperforms a range

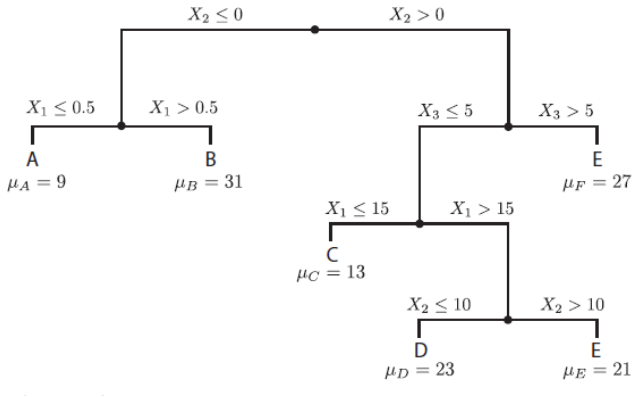

Figure 1: An exemplary regression tree from Montgomery et al. (2015).

of other tree-based ensemble methods on a collection of 42 benchmark datasets from the machine-learning literature.

4 Study Design

While the former sections have motivated the use of machine-learning approaches in election forensics, reflected on data and measurements, and outlined a particular method that may be deployed to detect electoral fraud, this last section presents some thoughts on the underlying study design. In general, we aim to (i) comprehensively evaluate the performance of an array of algorithms, (ii) evaluate them against traditional forensic techniques, and (iii) investigate whether they show to be of use for the analysis of actual voting returns. Therefore, this study comprises two general steps.

At first, by training algorithms on synthetic data structures as outlined above, a range of machine-learning approaches can be compared to each other and classical techniques of the forensic literature. Since we are not interested in particular scenarios but their performance on wide range of potential tasks, Table 4 sketches a preliminary simulation design varying the number of electoral units that in each set of voting returns, the proportion of tainted vote counts, the degree of manual interference, the underlying voter support for each of the two competing parties, as well as turnout levels per electoral unit. Finally, simulation experiments are designed for scenarios in which data is analyzed on its lowest level of aggregation and in scenarios in which data is aggregated into clusters after manual manipulations have been performed.

From this table, it becomes clear how the performed study is designed to tackle a variety of questions. First, this study is suitable to evaluate how successful trained algorithms are in identifying and quantifying mechanisms of fraud. Second, it can answer a variety of general questions about the use of machine-learning techniques in election forensics, such as how many electoral units researchers are in need of in order to reliably apply these methods or if predictive models are useful even if underlying voting returns consists of aggregated data that might have blurred the lines between clean and tainted units of analysis.

Additionally to evaluating my learners on these simulated data sets, this project will apply the

most successful algorithms to selected real voting returns in order to demonstrate their practical use.

Table 4: Varying factors in simulation design.

	Description	Denotion	Range
Data generating methodology			
	Electoral units	n	200, 300, ..., 10000
	Proportion of tainted vote counts	p	0.05, 0.1, ..., 0.4
	Degree of interference	λ	0.01, 0.02, ..., 0.2
	Vote support ratios for parties	a_j	2:1, 1.33:1, 1:1
	Turnout per electoral unit	t_i	0.4, 0.5, 0.6, 0.7
	Aggregated vote returns	—	Yes, No
Statistical learners			
	Algorithms for classification/prediction	—	Bayesian additive regression trees, Random forests, Naive Bayes, Feed-forward neural networks

References

Beber, B. and Scacco, A. (2012), 'What the Numbers Say: A Digit-Based Test for Election Fraud', *Political Analysis* **20**, 211–234.

Cantú, F. and Saiegh, M. S. (2011), 'Fraudulent Democracy? An Analysis of Argentina ' s Infamous Decade Using Supervised Machine Learning', *Political Analysis* **19**(4), 409–433.

Chipman, H. A., George, E. I. and McCulloch, R. E. (2012), 'BART: Bayesian additive regression trees', *Annals of Applied Statistics* **6**(1), 266–298.

Klimek, P., Yegorov, Y., Hanel, R. and Thurner, S. (2012), 'Statistical detection of systematic election irregularities', *PNAS* **109**(41), 16469–16473.

Levin, I., Pomares, J. and Alvarez, R. M. (2016), Using Machine Learning Algorithms to Detect Election Fraud, *in* 'Computational Social Science', Cambridge University Press, Cambridge, pp. 266–294.

Mebane, W. R. and Klaver, J. (2015), Election Forensics: Strategies versus Election Frauds in Germany.

Montgomery, J. M., Olivella, S., Potter, J. D. and Crisp, B. F. (2015), 'An informed forensics approach to detecting vote irregularities', *Political Analysis* **23**(4), 488–505.

Myagkov, M., Ordeshook, P. C. and Shakin, D. (2009), *A Forensics Approach to Detecting Election Fraud*, Cambridge University Press, pp. 1–289.

Rozenas, A. (2017), 'Detecting Election Fraud from Irregularities in Vote-Share Distributions', *Political Analysis* **25**, 41–56.