Universität Osnabrück
Fachbereich Humanwissenschaften
Institute of Cognitive Science

Masterthesis

Systematic Evaluation and Optimization of Outbreak-Detection Algorithms Based on Labeled Epidemiological Surveillance Data

Rüdiger Busche
968684
Master's Program Cognitive Science
November 2018 - April 2019

First supervisor: Dr. Stéphane Ghozzi
Robert Koch Institute
Berlin

Second supervisor: Prof. Dr. Gordon Pipa
Institute of Cognitive Science
Osnabrück

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

city, date	signature

Abstract

Indicator-based surveillance is a cornerstone of Epidemic Intelligence, which has the objective of detecting disease outbreaks early to prevent the further spread of diseases. As the major institution for public health in Germany, the Robert Koch Institute continuously collects confirmed cases of notifiable diseases from all over the country. Each week the numbers of confirmed cases are evaluated for statistical irregularities. Theses irregularities are referred to as *signals*. They are reviewed by epidemiologists at the Robert Koch Institute and state level health institutions, and, if found relevant, communicated to local health agencies. While certain algorithms have been established for this task, they have usually only been tested on simulated data and are used with a set of default hyperparameters. In this work, we develop a framework to systematically evaluate outbreak detection algorithms on labeled data. These labels are manual annotations of individual cases by experts, such as records of outbreak investigations. For this purpose we use a custom score to summarize the quality of algorithms in a single number. We optimize the joint hyperparameter space of all algorithms using a combination of Bayesian and multi-armed bandit strategies. We find that generalizable improvements of outbreak detection performance can be obtained using hyperparameter optimization. However, the performance on real data of all tested algorithms is rather weak compared to what would be expected from simulation studies. This indicates that outbreak detection on real data is a hard problem and that there is room for improvement by designing new outbreak detection algorithms. With the establishment of an evaluation framework we enable a fair comparison of outbreak detection algorithms on real world data. This also opens up the door to comparing classical outbreak detection methods with machine learning-based and anomaly detection methods from other fields.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Motivation and Objective

Disease outbreaks pose a major risk to public health. Through timely detection and appropriate reaction their consequences can be mitigated. The processes of collecting and evaluating information relevant to public health risks as well as the mechanisms for reacting to them are summarized by the *World Health Organization* (WHO) in the Early Warning and Response framework [1] (EWAR). The WHO defines EWAR as

> the organized mechanism to detect as early as possible any abnormal occurrence or any divergence from the usual or normally observed frequency of phenomena.

The ability to detect public health risks early is included in definitions of *surveillance* and *Epidemic Intelligence* (EI).

The WHO defines surveillance in the International Health Regulations [2] as

> the systematic on-going collection, collation and analysis of data for public health purposes and the timely dissemination of public health information for assessment and public health response as necessary.

and Epidemic Intelligence as

> the systematic collection, analysis and communication of any information to detect, verify, assess and investigate events and health risks with an early warning objective. [1]

These two rather synonymous definitions are further subdivided into *Event-Based Surveillance* and *Indicator-Based Surveillance* (IBS). Event-Based Surveillance is concerned with unstructured data obtained from heterogenous sources at varying frequencies. In contrast Indicator-Based Surveillance deals with structured data from reliable sources. Often the data collection process in Indicator-Based Surveillance

is performed by the same organization that is responsible for the surveillance [1]. Indicator-Based Surveillance is specifically defined as

> the systematic (regular) collection, monitoring, analysis and interpretation of structured data, i.e. of indicators produced by a number of well-identified, mostly health-based, formal sources. [1]

A surveillance system in operation will generate *signals*. A signal again is defined as

> data and/or information considered by the Early Warning and Response system as representing a potential acute risk to human health. Signals may consist of reports of cases or deaths (individual or aggregated), potential exposure of human beings to biological, chemical or radiological and nuclear hazards, or occurrence of natural or man-made disasters. Signals can be detected through any potential source (health or non-health, informal or official) including the media. Raw data and information (i.e., untreated and unverified) are first detected and triaged in order to retain only the one pertinent to early detection purposes i.e. the signals. Once identified signals must be verified. When it has been verified, a signal becomes an "event".

In this work we focus on IBS based on confirmed case counts. This is the most conservative form of IBS as it only includes information of high confidence, i. e. cases that conform to reference definitions and are often corroborated by laboratory investigation. Moreover, it is the main form of surveillance currently employed at a national scale for a wide range of infectious diseases by the Robert Koch Institute (RKI), as regulated by the 2001 Infection Protection Act [3, 4].

The objective of this work is to empirically evaluate the quality of outbreak detection algorithms in the setting of the surveillance system of mandatory notifiable diseases at the RKI.

While many studies [5–8] have compared the performance of standard outbreak detection methods they usually exhibit the following limitations

- The data sets used for evaluation is either completely simulated or the outbreak labels are created algorithmically.
- In most studies, reporting delay is not incorporated in the data. Instead data is treated as if all data would be known in the moment outbreak detection is applied.

- The studies collect different metrics, such as sensitivity and specificity, and interpret them, but do not provide a definite ranking of algorithms.
- Default hyperparameters are used or only a very coarse grid search is conducted, leaving room for unseen performance in unexplored hyperparameter space.

We try to overcome these limitations with the following approaches:

- By comparing the signals generated by common outbreak detection algorithms on data from the RKI's surveillance system with expert labeled outbreaks, we assess how well the current system actually performed in the past and how well other algorithms would have performed in place of it.
- By applying the outbreak detection only to the data available at the point in time when the outbreak had to be detected, we obtain a more realistic estimate of algorithm performance.
- By defining a custom scoring rule we summarize algorithm performance in a single number.
- Through hyperparameter optimization we ensure a fair comparison between algorithms and investigate if hyperparameter optimization achieves generalizable improvements in the outbreak detection setting.

By evaluating the signal detection aspect of the RKI's surveillance system we contribute to complying with WHO guidelines for communicable disease surveillance and response systems [9], which demand that surveillance systems are evaluated for achievement of surveillance objectives and possible improvements periodically.

1.2 The Surveillance System at the RKI

The Surveillance System at the RKI [10] collects confirmed cases of 88 notifiable diseases, including information about the patient (age, sex, etc.) and the infection (place, time, pathogen subgroup, etc.).

When a patient goes to see a physician and a notifiable disease is diagnosed, a report is sent to the local public health agency. Simultaneously, a probe is given to a laboratory. The laboratory could be part of the physician's practice or an independent institution. If the laboratory can corroborate the diagnosis, it also

reports to the local public health agency. Only when the local agency has the confirmation from both the physician and the laboratory, it calls the reported case a *confirmed* case and reports it to the state public health agency. State agencies pass the information on to the RKI without further interference. Figure 1.1 summarizes the reporting pathway. While this reporting pathway ensures that only actual cases are reported as confirmed cases, it leads to a significant delay between first diagnosis and having the data available at the RKI

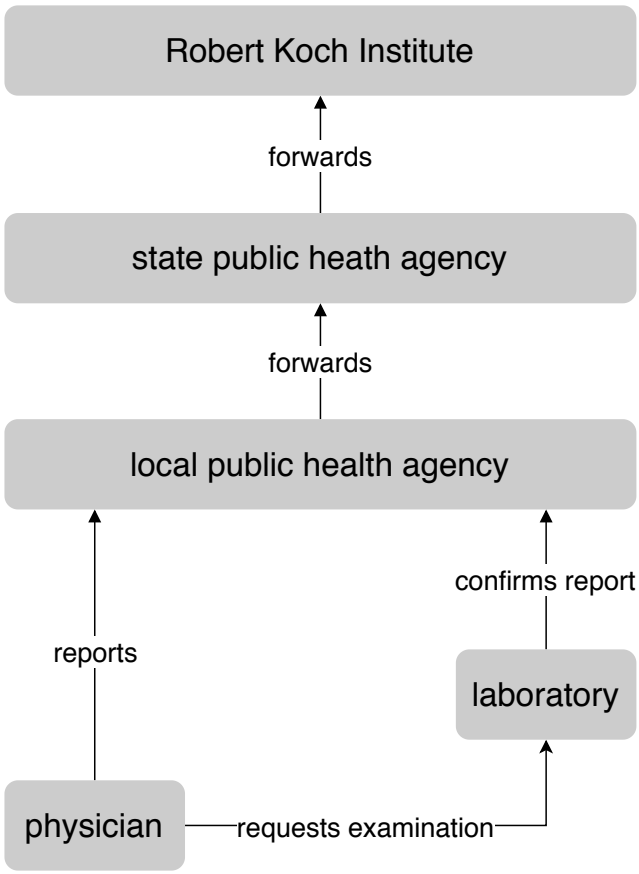


Figure 1.1: Schematic description of the German reporting system. Reproduced from Salmon [11].

At the RKI, time series data are created through weekly aggregation of case counts according to some combination of attributes, so called *filter combinations*. Therefore the same case could be included in several time series. For example, the case of a woman infected with *Salmonella* in Munich, would be included in female *Salmonella* cases in Munich, *Salmonella* cases in Munich and *Salmonella* cases in Bavaria.

The FarringtonFlexible algorithm [12] is used to look for signals in each of those time series. The detected signals are stored in a separate database and presented to epidemiologist at the RKI as well as local health agencies in the form of a dynamic report [13], (Figure 1.2). In this application, users can decide to only look at signals at more general levels of filter combinations, e.g. whole Bavaria, or at more specific levels, e.g. women in Munich [10].

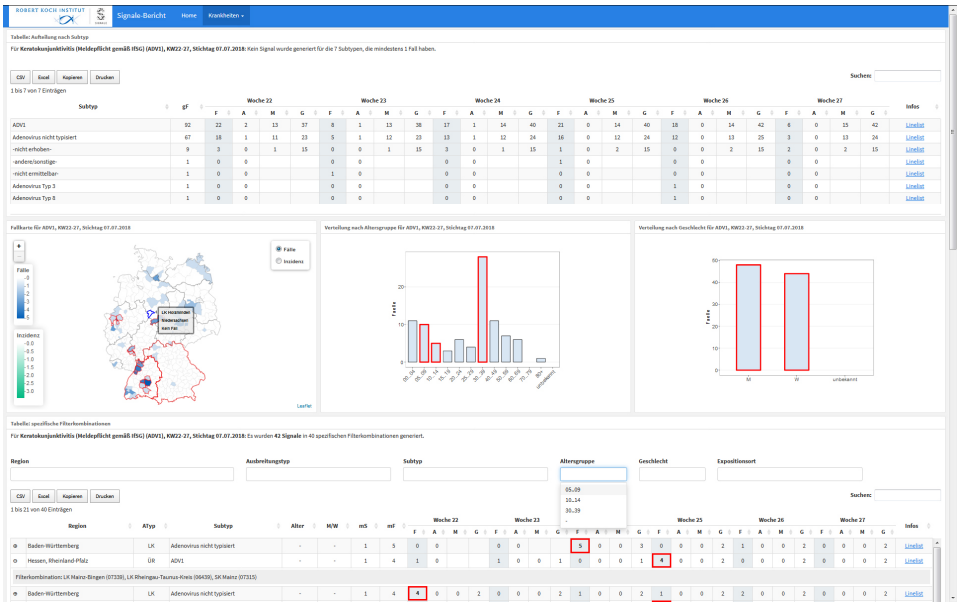


Figure 1.2: Screenshot of the dynamic signal report send weekly to epidemiologist at the RKI and state public health agencies.

1.3 Formalizing the Outbreak Detection Problem

In this section we explore different ways of formalization the problem of outbreak detection. Even though the term outbreak is used routinely, no commonly accepted precise definitions exists. The WHO gives the following definition [14]:

Definition 1.1. *Outbreak according to WHO* A disease outbreak is the occurrence of disease cases in excess of normal expectancy. The number of cases varies according to the disease-causing agent, and the size and type of previous and existing exposure to the agent.

While such a definition is very general, it is also very imprecise, because it does not address the question what defines “normal expectancy”. This opens the gates

for many possible quantitative outbreak definitions, leading to a multitude of inconsistent definitions [15].

What seems closer to epidemiological practice is to assign cases to an outbreak if they share a common origin. An example of this can be found in the outbreak definitions for Norovirus of the Australian Department of Health [16].

Definition 1.2. *Epidemiological outbreak* Two or more cases of common etiology.

As Brady *et al.* [15] note, an outbreak in epidemiological terms does not necessarily constitute a situation that is a threat to public health. Therefore, only outbreaks that exceed the normal control capacities in operation should be considered for triggering additional public health responses. A definition for outbreaks relevant to surveillance could therefore be:

Definition 1.3. *Surveillance relevant outbreak* A situation in which the high risk of acquiring a disease is present in the population such that it warrants intervention by a public health institution.

Even though the epidemiological and the surveillance relevant definition might often overlap, it is possible that only the epidemiological definition is met. For example, it might be possible to trace several cases to the same source, but these cases do not have any risk of transmitting the disease further. For example, a couple living in social isolation far out in the mountains, where one partner infects the other. In this case the epidemiological definition would be met, but the surveillance definition would not be met. In our understanding, the surveillance definition subsumes the epidemiological definition, such that a surveillance relevant outbreak always has an epidemiological origin. We emphasize that while the epidemiological definition might be more precise, the surveillance definition provides a working definition for the practitioner.

Irrespective of the exact definition an outbreak is always characterized by at least two cases that are assigned to the same outbreak label. In our case this labeling is provided by experts as part of the surveillance system at the RKI system. In the following we give a mathematical formalization of the problem.

Let $C = \{(\mathbf{x}_i, \tau_i, y_i)\}_{i=1}^{N}$ be the set of cases. Each case c_i consist of a feature vector $\mathbf{x}_i \in X$ that contains the disease, information about the patient such as age, sex and location, a real timepoint $\tau_i \in \mathbb{T}$ which is the time of infection and an outbreak label y_i with

$$y_i = \begin{cases} 0 & \text{if } c_i \text{ is not element of an outbreak} \\ j \in \mathbb{N} & \text{if } c_i \text{ belongs to outbreak } j \end{cases}$$

A general formulation for outbreak detection is to map unlabeled cases onto their outbreak labels

$$f : X \times \mathbb{T} \to \mathbb{N}_0 \tag{1.1}$$

This resembles the classical cluster problem formulation with a variable number of cluster centers as in DBSCAN [17].

Even though this case based formulation seems natural given the nature of diseases, they are not employed in the world of surveillance algorithms and also go beyond the scope of this work. While human experts actually label individual cases as belonging to individual outbreaks, surveillance algorithms usually work on regular spaced aggregated time series of case counts. Let $\mathbf{x} = (x_1, \dots x_T)$ be such a time series with entries at regularly spaced, discrete timepoints t. An entry x_t of that time series is defined as

$$x_t = \left|\{c_i \mid t \leq \tau_i < t + 1\}_{i=1}^{N}\right| \tag{1.2}$$

with $t + 1$ denoting the next time point in the time series.

Based on this transformation we can view the problem as a *sequential supervised learning problem* [18], in which the sequence of counts is paired with a sequence of outbreak labels $(\mathbf{x}, \mathbf{y})$, with $\mathbf{x} = (x_1, \dots, x_T), x_i \in \mathbb{N}_0$ and $y_i \in \mathbb{B}$. For each timepoint t a boolean label is assigned, corresponding to whether there were outbreak cases present in the aggregation time interval. We will label this problem *time point classification problem*. This is the standard formulation of common surveillance algorithms. Alternatively, the number of outbreak cases in the aggregation time interval could be used as labels, in which case we would call it a *time point regression problem*.

The time point formulation can be extended into a time series formulation by dividing the time series $\mathbf{x}$ into smaller time series and assigning the label of the last time point to the whole time series. Thus a data set $\{(\mathbf{x}_j, y_j)\}_{j=1}^{T}$ is obtained. This formulation is especially useful for incorporating reporting delay. That means that

the information at time point $t = j$ can be quite different depending on whether j is relatively recent, e. g. $j = T$ or already some time in the past. This is due to the fact that information arrives sometimes slowly in epidemiological surveillance systems. We will call this problem formulation a *time series classification problem*. It is the main problem formulation used in this work.

2 Methods

2.1 Outbreak Detection Algorithms

In this section we review the outbreak detection techniques evaluated in this work. Our aim is to sort the different approaches into general categories. For a more detailed description of the exact statistical methodologies we refer to Unkel *et al.* [19] and Salmon *et al.* [20].

What all algorithms have in common is that they can be viewed as *semi-supervised* techniques from a machine learning or anomaly detection perspective [21]. All algorithms fit historic data, assuming that they represent the normal state of the system. Having fitted the data, an estimate for the case counts of the current week is computed. This estimate is compared to the number of cases reported in the current week. If the observed case count exceeds the expected number by some threshold, an alarm is raised. Most algorithms in fact compute a predictive distribution for the estimated number of case counts and raise an alarm if the actual number exceeds a certain quantile of this distribution.

We choose to evaluated all algorithms available in the R surveillance package [20] that yielded themselves to the problem formulation defined in Section 1.3 and did not have prohibitively large run-times that would make the application of hyperparameter optimization difficult. All outbreak detection algorithms evaluated in this work are summarized in Table 2.1.

Moreover, all algorithms evaluated in this work focus on *univariate* time series of counts. While extensions for multivariate time series and incorporation of spatial data exist [19], they are not considered in this work.

2.1.1 Window-based Approaches

The simplest form of outbreak detection algorithms are window-based approaches. For them the expectation for the current week is computed from a moving window of fixed size. For example the `EarsC1` algorithm, computes its predictive distribution

based the mean and standard deviation of the last seven timepoints, using a normal distribution.

Because of the short time interval considered, these approaches are naturally insensitive against seasonality and trend. However, recent outbreaks can contaminate the data, reducing the sensitivity of the algorithms.

This category includes the Ears-family [22], CDC [23] and the RKI [20] algorithm.

2.1.2 GLM-based Approaches

Approaches based on Generalized Linear Models (GLMs) form a popular group of outbreak detection algorithms. They compute a predictive distribution for the current week based on fitting a GLM to previous data. An alarm is raised if the current observation is unlikely under the predictive distribution controlled by some α value. Often Poisson or Negative Binomial models are used to do justice to the count nature of the data. Moreover, terms to accommodate seasonality and trend are often incorporated as well. GLM-based approaches included the classical Farrington algorithm [24] and its more recent extension [12].

2.1.3 Cusum-based Approaches

Both window-based and GLM approaches have the downside that they only incorporate evidence from the current week. Larger outbreaks that build up slowly could therefore easily be missed. Cusum-based approaches are inspired by models from statistical process control [25] and incorporate evidence from previous timepoints. Instead of computing a predictive distribution, evidence that observed case counts do originate from an epidemic is accumulated until a certain threshold is exceeded and an alarm is raised. Then the sum is reset.

Cusum-based approaches include the Cusum [26], generalized likelihood ratio methods based on Poisson [27] or negative binomial distributions [28] and the OutbreakP method [29].

Table 2.1: List of evaluated outbreak detection algorithms.

algorithm	category	reference
EarsC1	window-based	Hutwagner *et al.* [22]
EarsC2	window-based	Hutwagner *et al.* [22]
CDC	window-based	Stroup *et al.* [23]
RKI	window-based	Salmon *et al.* [20]
Farrington	GLM-based	Farrington *et al.* [24]
FarringtonFlexible	GLM-based	Noufaily *et al.* [12]
Cusum	Cusum-based	Rossi *et al.* [26]
GLRPoisson	Cusum-based	Höhle [27]
GLRNegativeBinomial	Cusum-based	Höhle & Paul [28]
OutbreakP	Cusum-based	Frisén *et al.* [29]
Bayes	other	Höhle [30]

2.2 Hyperparameter Optimization

Hyperparameter optimization is the field of algorithmically optimizing parameters that are usually fixed *before* the training of a model [31]. We employ a state-of-the-art approach called BOHB [32], to optimize the joint hyperparameter space of each algorithms. BOHB combines two approaches to hyperparameter optimization: Bayesian Optimization and Hyperband.

Machine learning algorithms are parameterized by a set of hyperparameters $\mathbf{x} \in \mathcal{X}$ before learning. The set of possible hyperparameters $\mathcal{X}$ can be characterized as a tree-structured generative process involving both discrete and continuous spaces. For example, a categorical variable might be the procedure to fit a time trend. Depending on the procedure other continuous parameters that influence the trend fitting process are added. The loss defined for the task at hand induces an evaluation function $f : \mathcal{X} \to R^+$ that maps a hyperparameter configuration to the validation loss attained after training. The goal of hyperparameter optimization is to find a configuration x^* that minimizes f, i. e. $\mathbf{x}^* = \text{argmin}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.

As the evaluation of f involves training of the model, it is too expensive to be performed exhaustively over large hyperparameter spaces. The idea of *model-based hyperparameter optimization* is to instead *model* f based on observed pairs of hyperparameters $\mathbf{x}$ and the loss l achieved with these hyperparameters $D = \{\mathbf{x}_i, l_i\}_{i=1}^N$. This way hyperparameter optimization can be formulated as another learning problem. In each iteration a model M is fitted to the data and subsequently used to

sample new promising hyperparameter configuration (cf. Algorithm 1).

Algorithm 1: Pseudo-code for generic model-based hyperparameter optimization. Adapted from Bergstra *et al.* [31].

Data: configuration space $\mathcal{X}$, evaluation function f, model M, initial model M_0
Result: best configuration x^*
$D \leftarrow \emptyset$;
for $t \leftarrow 1$ *to* T **do**
  sample promising configuration x^* from model M_{t-1};
  $l_t \leftarrow f(x^*)$; `// evaluate objective function`
  $D \leftarrow D \cup (x^*, l_t)$; `// add new data point to dataset`
  $M_t \leftarrow M(D)$; `// refit model to new dataset`
end

The idea behind Bayesian optimization is to model the probability density $p(f|D)$. Based on this, an acquisition function a is constructed, that is maximized to find new promising configurations. Expected improvement is a common choice for the acquisition function. Let l^* denote the lowest validation loss observed so far.

$$a(\mathbf{x}) = \int \max(0, l^* - f(\mathbf{x}))\mathrm{d}p(f|D) \tag{2.1}$$

Tree-structured Parzen Estimators (TPE) [31] exploit this choice by directly optimizing expected improvement instead of modeling $p(f|D)$. TPEs use kernel density estimators to model the two distributions

$$l_{\mathrm{TPE}}(\mathbf{x}) = p(f(\mathbf{x} < l^*|\mathbf{x}, D)) \tag{2.2}$$

$$g_{\mathrm{TPE}}(\mathbf{x}) = p(f(\mathbf{x} > l^*|\mathbf{x}, D)) \tag{2.3}$$

Then the next configuration is selected as $\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x}\in\mathcal{X}} \frac{l_{TPE}(\mathbf{x})}{g_{TPE}(\mathbf{x})}$. As Bergstra *et al.* [31] proved this is equivalent to optimizing expected improvement.

Hyperband [33, 34] on the other hand is an extension of random search. To speed up the process of finding good configurations, it evaluates configurations at first on a smaller *budget*, replacing the evaluation function with a budgeted version $\tilde{f} : \mathcal{X} \times \mathbb{N} \to \mathbb{R}^+$. The budget can be the number of training epochs, or a smaller proportion of the dataset. The idea behind this is that performance on smaller tasks

is indicative of the performance on the entire tasks. A large number of random configurations is thus evaluated on a small budget. Only the best k configurations will be advanced to the next round, where they are evaluated on a bigger budget. This procedure called `SuccessiveHalving` is performed until the maximum budget is reached (cf. Algorithm 2).

Algorithm 2: Pseudo-code for Hyperband. Adapted from Falkner *et al.* [32].

Data: minimum budget $b_{\min}$, maximum budget $b_{\max}$, budget progression parameter η, number of iterations i
Result: best configuration x^*
$s_{\max} \leftarrow \left\lfloor \log_\eta \frac{b_{\max}}{b_{\min}} \right\rfloor$;
while $i > 0$ **do**
  for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
    sample $n = \lceil \frac{s_{\max}+1}{s+1} \cdot \eta^s \rceil$ candidate configurations C;
    `// start SuccessiveHalfing subroutine`
    $b \leftarrow \eta^s \cdot b_{\max}$;
    while $b \leq b_{\max}$ **do**
      $L \leftarrow \{f(c, b) \mid c \in C\}$; `// evaluate losses on budget` b
      $k \leftarrow \left\lfloor \frac{|C|}{\eta} \right\rfloor$;
      $C \leftarrow \text{top}(C, L, k)$; `// only keep the` k `best configurations`
      $b \leftarrow \eta \cdot b$; `// increase budget`
    end
    $i \leftarrow i - 1$;
  end
end

BOHB unites these two approaches by replacing the random sampling in Hyperband with a TPE model. Starting with random sampling, a TPE model is build once enough configurations are available to learn from. To keep diversity in the sampled configurations BOHB still uses a fraction of random configuration in each iteration even after the TPE model is available.

2.3 Implementation

Python

Most code for this work was written in Python [1]. Python is an interpreted, dynamically typed language originally created by Guido van Rossum and developed by an open source community. Key features of Python are its flexibility, its simplistic syntax and its interoperability with C. While Python is strictly object-oriented in its implementation and has wide support for object-oriented programming paradigms, it also allows for procedural or functional programming styles. Python was an excellent choice for this work, as it offers many mature packages for machine learning, hyperparameter optimization and experiment tracking (cf. the following subsections). Moreover, it allows for cleanly engineered software while being fast for prototyping. For this work, Python was used in version 3.7.

R surveillance package

Reference implementations for all discussed outbreak detection algorithms are taken from the R surveillance package [30].

epysurv

To be able to use the outbreak detection algorithms from the R surveillance package in Python, we wrote a wrapper for the library called `epysurv`. It provides a `scikit-learn`-like interface [35] to all outbreak detection algorithms. As all `scikit-learn` estimators, the algorithms provide a `fit` and a `predict` method. The `fit` method has to be provided with a regularly time-indexed `DataFrame` [36] that contains the case and outbreak case counts for each time point. When calling `predict`, a time-indexed `DataFrame` has to be provided containing only the case counts. For each time point in the `DataFrame` a prediction is made. The time points provided to `predict` should lie strictly in the future of the time points provided to `fit`. This interface was inspired by the `prophet` library [37]. For each algorithm, an interface based on single arrays for the time point problem formulation and generator based interfaced for the time series problem formulation is provided (cf. Section 1.3). The generators provided have to yield a `DataFrame`-label pair at each iteration. Using generators allows to utilize the time series problem in a memory efficient manner. In addition to the outbreak detection algorithms, `epysurv` provides some

[1] `https://www.python.org/`

utilities for data handling, plotting, and simulation. As part of this work, `epysurv` was released as an open source package[2] and can be installed via `conda`[3].

Sacred

Sacred [38] is an experiment tracker that is designed to make machine learning experiments transparent and reproducible. We used it to keep track of different runs, i. e. hyperparameter optimizations and the performance of single hyperparameter configurations.

Using Sacred we could easily store metrics associated with different hyperparameter configurations and subsequently analyze them. As Sacred stores source code alongside parameter configurations, we ensure reproducibility of our experiments.

Additionally, we created a package called `incense`[4] to easily query and display the logged results from Sacred. `incense` was open sourced as part of this work.

HpBandSter

To make use of the BOHB algorithm, we used the `HpBandSter` package, in which BOHB was originally implemented. In addition to the implementation of BOHB, `HpBandSter` allows for the distributed execution of the hyperparameter optimization across different cores or even computing clusters.

2.4 IT Infrastructure at the RKI

The RKI has a rather conservative IT-infrastructure, which uses Windows 7 for developer machines. We choose to develope the software on a private Linux machine instead and only work with synthetic data during the development process. This made it possible to easily open source `epysurv`, because it was not entangled with sensitive data. Data was prepared on the Windows machine and transferred to the RKI's computing cluster. The code was transferred to the computing cluster using a remotely hosted git repository. Computation was performed on the computing cluster and results were recorded using a MongoDB instance running locally on the computing cluster using Sacred. The directory were the MongoDB stored its data was then mounted via the network on the Windows machine and analyzed there.

[2] `https://github.com/JarnoRFB/epysurv`
[3] `https://docs.conda.io/en/latest/`
[4] `https://github.com/JarnoRFB/incense`

Thus sensitive data never left the RKI system, while software development could be performed in a developer friendlier environment.

2.5 Scoring

Outbreak detection is different from standard binary classification. It is characterized by high class imbalance. Moreover, each classification decision is associated with a case count and timeliness information. It might be more important to recognize large outbreaks and it might be of more value to recognize them early.

To meet these special requirements we designed a custom score that incorporates the case count. It is defined as follows:

Let $\mathbf{x}$ be an epidemiological time series consisting of endemic cases $\mathbf{b}$ (background) and epidemic cases $\mathbf{o}$ (outbreak). Particularly, at any point in time t the number of cases is the sum of endemic and epidemic cases

$$x_t = b_t + o_t$$

In the score we weight true positives and false negatives by the number of outbreaks cases at the particular timepoint. This is motivated by the fact that this number of cases is either identified or missed as belonging to an outbreak. False positive predictions are punished by the mean number of outbreak cases in an outbreak week. This is based on the assumption that a falsely raised alarm would cause on average that much preventive action in the local health agency. True negatives are not considered, with the idea to not reward specificity to strongly.

Let $\mathbf{a}$ be the boolean time series of alarms raised by an outbreak detection model. Then we define the score s associated with alarms and actual outbreaks as

$$s(\mathbf{o}, \mathbf{a}) = \frac{\sum_t \mathbb{1}(o_t = a_t \wedge a_t = 1) \cdot o_t - \mathbb{1}(o_t \neq a_t \wedge a_t = 0) \cdot o_t - \mathbb{1}(o_t \neq a_t \wedge a_t = 1) \cdot \bar{\mathbf{o}}}{\sum_t o_t} \tag{2.4}$$

This score has some desireable properties, which can be illustrated using some simple models. Let m^* be the model that predicts all outbreaks correctly and never raises a false alarm, i. e. $\mathbb{1}(o_t > 1) = a_t \; \forall \; t$. We write $s(m)$ to denote the score that a model achieves, if the score can be calculated independently of actual data. It follows that m^* achieves the best possible score of 1.

$$s(m^*) = \frac{\sum_t \mathbb{1}(o_t = a_t \wedge a_t = 1) \cdot o_t}{\sum_t o_t} \tag{2.5}$$

$$= 1 \tag{2.6}$$

$$\tag{2.7}$$

A model m_0 that will never predict an outbreak ($a_t = 0 \ \forall t$) on the other hand will only achieve a score of -1.

$$s(m_0) = \frac{\sum_t -\mathbb{1}(o_t \neq a_t \wedge a_t = 0) \cdot o_t}{\sum_t o_t} \tag{2.8}$$

$$= \frac{\sum_t -o_t}{\sum_t o_t} \tag{2.9}$$

$$= -1 \tag{2.10}$$

$$\tag{2.11}$$

A model m_∞ that always predicts an outbreak ($a_t = 1 \ \forall t$) will be assigned a score that depends on the ratio of outbreak and non-outbreak weeks. Let T be the total number of timepoints in the time series. Let k be the number of outbreak timepoints in the time series ($k = \sum_t \mathbb{1}(o_t > 0)$).

$$s(m_\infty) = \frac{\sum_t \mathbb{1}(o_t = a_t \wedge a_t = 1) \cdot o_t - \mathbb{1}(o_t \neq a_t \wedge a_t = 1) \cdot \bar{\mathbf{o}}}{\sum_t o_t} \tag{2.12}$$

$$= \frac{\sum_t \mathbb{1}(o_t = a_t \wedge a_t = 1) \cdot o_t}{\sum_t o_t} - \frac{\sum_t \mathbb{1}(o_t \neq a_t \wedge a_t = 1) \cdot \bar{\mathbf{o}}}{\sum_t o_t} \tag{2.13}$$

$$= \frac{\sum_t o_t}{\sum_t o_t} - \frac{(T-k) \cdot \bar{\mathbf{o}}}{\sum_t o_t} \tag{2.14}$$

$$= \frac{\sum_t o_t}{\sum_t o_t} - \frac{(T-k) \cdot \frac{1}{k} \sum_t o_t}{\sum_t o_t} \tag{2.15}$$

$$= 1 - \frac{T-k}{k} \tag{2.16}$$

$$= 2 - \frac{T}{k} \tag{2.17}$$

$$\tag{2.18}$$

A downside of this score is that it is undefined for time series without any outbreak cases, because of the normalization by the sum of all outbreak cases.

Another desireable property of the score would be to incorporate the timeliness of detected outbreaks. A tentative idea is to add the future cases of an outbreak to the score once an outbreak is detected. The implementation of adding a timeliness term to the score is left to future work.

3 Results and Discussion

3.1 Exploratory Analysis

Before the algorithms were optimized, several decisions had to be made regarding the structure of the dataset. In this section we describe the outcomes of analyses that informed the final structure of the dataset used in the optimization. The data that were analyzed are the case records for *Salmonella* and *Campylobacter*. We only show the results for *Salmonella* from the beginning of case recordings in 2001 up to early 2019 in detail. The results for *Campylobacter* are very comparable.

3.1.1 Reporting Delay

Because of the reporting pathway that a case needs to go through before it enters the case database (cf. Section 1.2), some time passes between a patient showing up at the doctor and the case record entering the database. This time lag is termed *reporting delay*. When aggregating individual cases into count time series, it is worth considering which variable to base the aggregation on and how to take the reporting delay into account. Two variables in the case data can be considered for this: `ReportingDate` and `OnsetOfDisease`. While `ReportingDate` marks the date at which the case was first entered into the database, `OnsetOfDisease` tries to give an estimate of when the patient initially got sick, i. e. before they visited the doctor. It seems more natural to use `OnsetOfDisease`, as outbreaks would be expected to appear when many people catch a disease. However, it turns out that 17% of cases are missing a value for the `OnsetOfDisease` variable. Moreover, a large proportion of cases is reported more than a week later than the onset of the disease (Figure 3.1). For this investigation, in which we only evaluate predictions for the current week, this would be very inconvenient as a large proportion of the data would have to be discarded when used for the task of prospective outbreak detection. We therefore decided to treat all cases as they occurred on the reporting date and not further consider the `OnsetOfDisease` variable. This also coincides with the current practice of the surveillance system operated at the RKI.

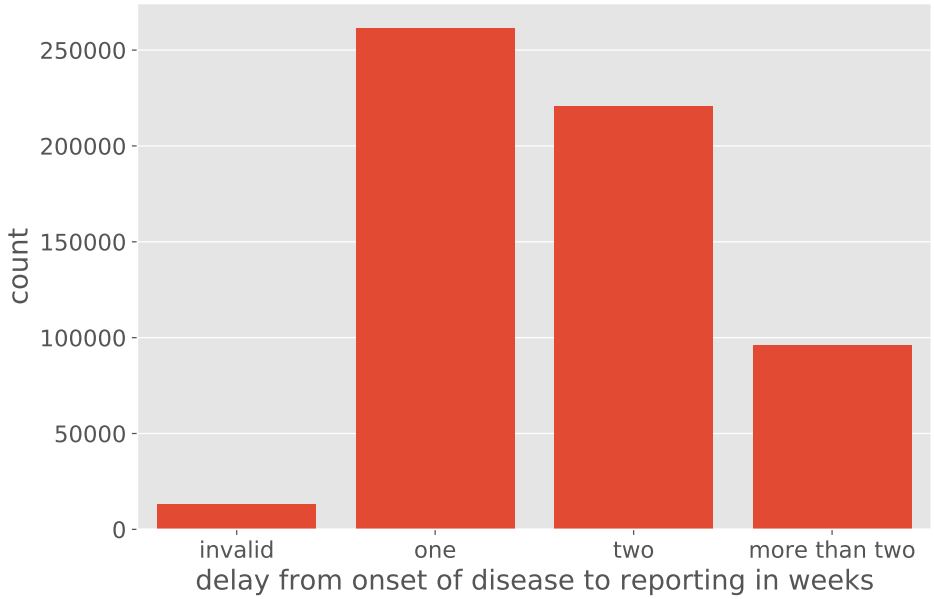


Figure 3.1: Counts of reporting delay for *Salmonella*. The label "invalid" corresponds to data points where the alleged onset of disease lies beyond the reporting date.

3.1.2 Labeling Delay

When aggregating case records into weekly count time series, a week is considered an outbreak week if it contains at least a single case that is part of an outbreak (cf. Section 1.3). Using semi-supervised outbreak detection algorithms can only make sense if this labeling is provided later than the original case records. If all outbreaks were known in the moment the cases are reported, their would indeed be no need for automatic outbreak detection. We therefore assessed the delay between the reporting of cases and their first time being labeled as belonging to an outbreak. In fact 42% of all outbreak cases are already known as outbreaks in the week they are reported and for 37% of all outbreaks at least one case is known to belong to the outbreak in the week the outbreaks starts (Figure 3.2). However, as the majority of outbreaks is unknown in the week they occur, it still makes sense to predict outbreaks. Now it would heavily distort the labeling if the outbreaks known in the current week would be discarded. Additionally, none of the established outbreak detection methods evaluated in this work is able to utilize the information about outbreaks directly. We therefore disregarded the fact that many outbreaks are known in the current

week. We leave the consideration about how to utilize this available information for future work.

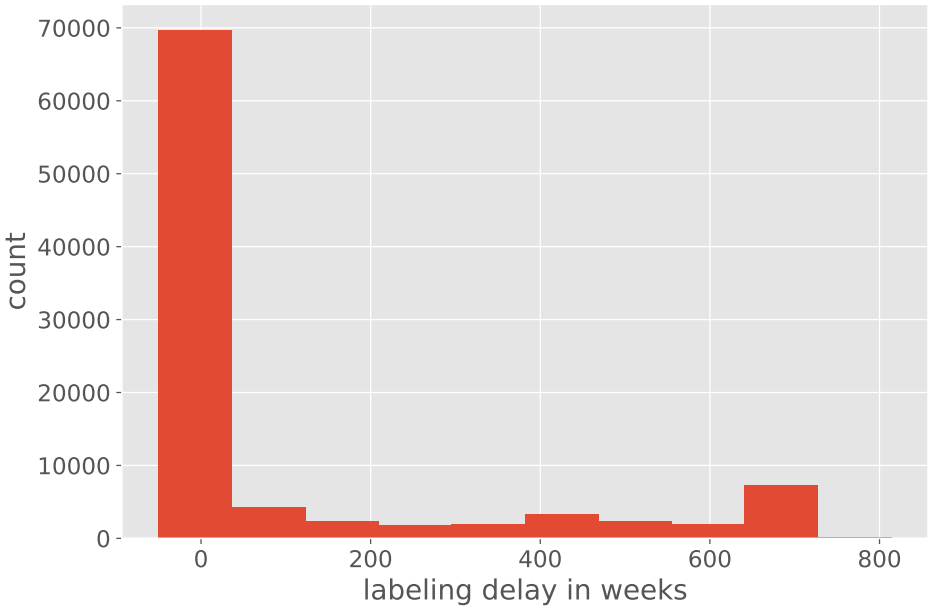


Figure 3.2: Histogram of labeling delay in weeks for *Salmonella*.

3.1.3 Outbreak spread

As described in Section 1.2 there are many possible filter combinations that can be used to aggregate case records into count time series. To minimize the computational load and the problem of multiple testing, we evaluated the spread of outbreaks over different filter values. One of the most important filters is the geographical location. The smallest spatial resolution of a case is a county. In the current system data is aggregated over single counties, several combinations of neighboring counties and the state level. We found that applying our labeling scheme to state level data makes relatively little sense. As any small outbreak causes a week to be labeled as an outbreak week, a large proportion of weeks ends up being called an outbreak. We therefore decided not to incorporate state level aggregation. Next we investigated how likely it is that outbreaks spread across multiple counties. We found that 95% of all outbreaks are restricted to a single county (Figure 3.3).

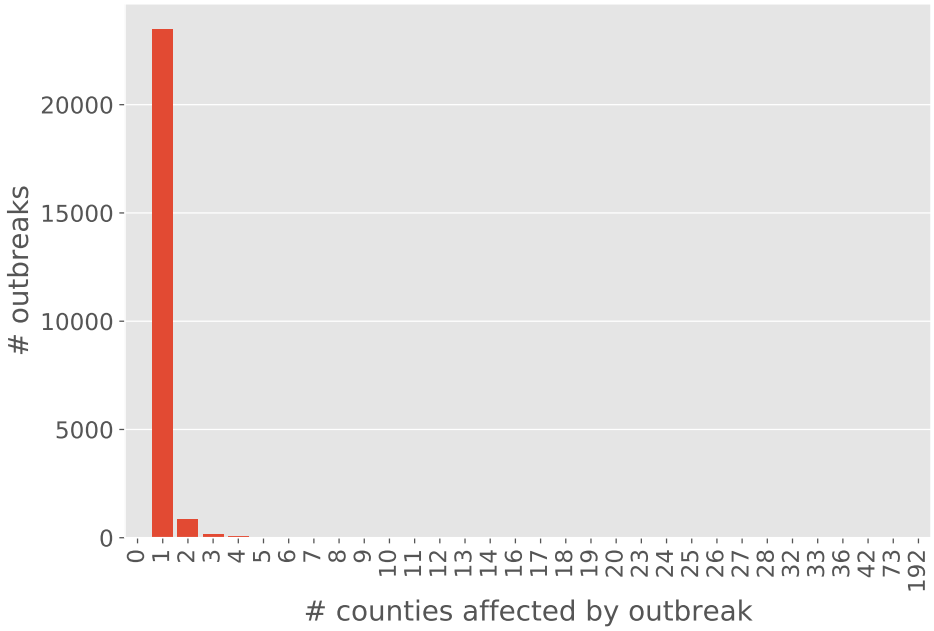


Figure 3.3: Number of outbreaks that spread across the respective number of counties for *Salmonella*.

We therefore decided to only aggregate by single counties, as this avoids evaluating the same data twice, while keeping all information that should be enough to detect the vast majority of all outbreaks.

Another filter variable is the pathogen subtype. Each disease recorded in the database can be caused by different subtypes of a pathogen. These subtypes are organized in a hierarchical fashion. We found that the majority of cases are caused by very few subtypes (Figure 3.4).

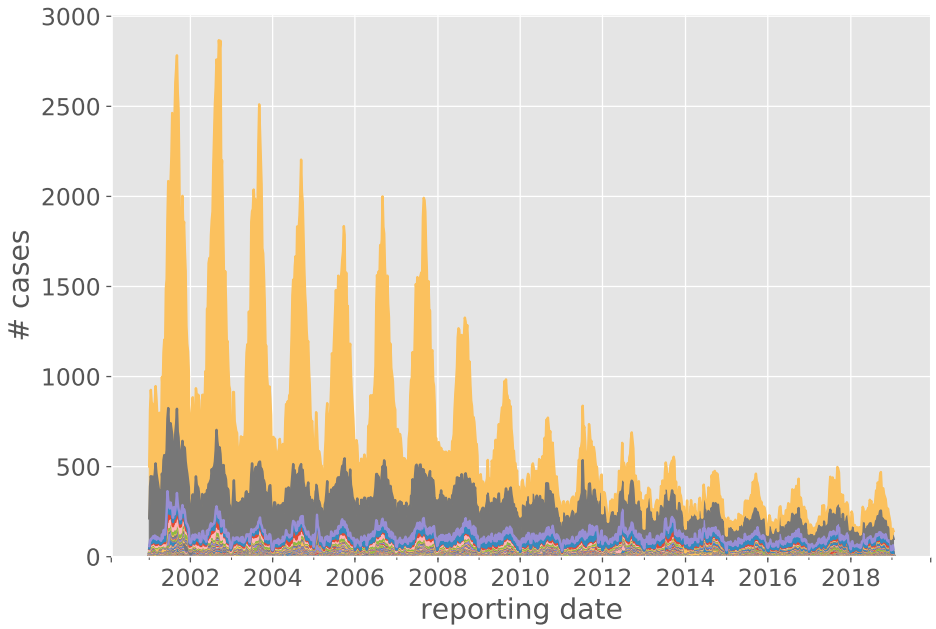


Figure 3.4: Number of cases per pathogen for *Salmonella*. Each colored area represent a different pathogen subtype.

As is the case for geographical information, in the surveillance system of the RKI, different levels of subtypes are considered for outbreak detection. Again, we evaluated how many distinct subtypes usually participate in an outbreak. We found that 96% of all outbreaks are caused by a single pathogen subtype (Figure 3.5).

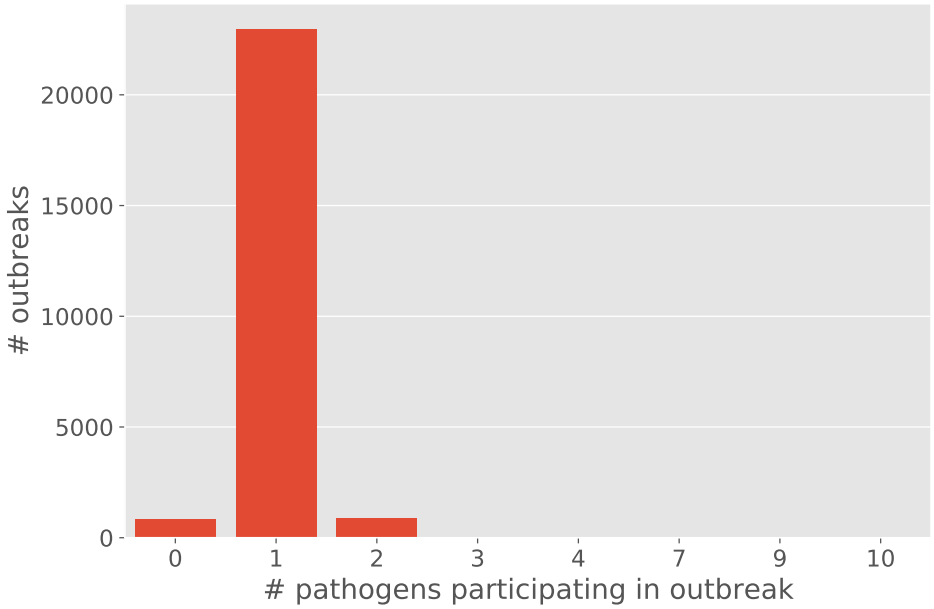


Figure 3.5: Number of outbreaks caused by different numbers of pathogens subtypes for *Salmonella*.

We therefore decided to only aggregate by single pathogen subtypes, for the same reasons as for aggregating only by single counties.

3.1.4 Timing of Case Reportings

In the beginning we considered the possibility of something else than weekly aggregation across the time axis, e. g. sliding window or exponential weighting. However, judging from the distribution of weekdays it seems that most cases are reported on Mondays (Figure 3.6). This gives rise to the assumption that relevant cases are collected throughout the week in local health agencies and then collectively send on Monday. Because of that, anything else than weekly aggregation does not really make sense, as it would only artificially distort the data. Weekly aggregation seems to be the finest resolution the data yields itself. As a good sign for the early reporting of cases, we find that the timing of reporting has changed over the course of time. Considering only the cases from 2015 onwards, the distribution of reporting timings is much more balanced, with the majority of cases still being reported on Mondays (Figure 3.7). To be consistent with the majority of older data, we decided to stick with weekly aggregation.

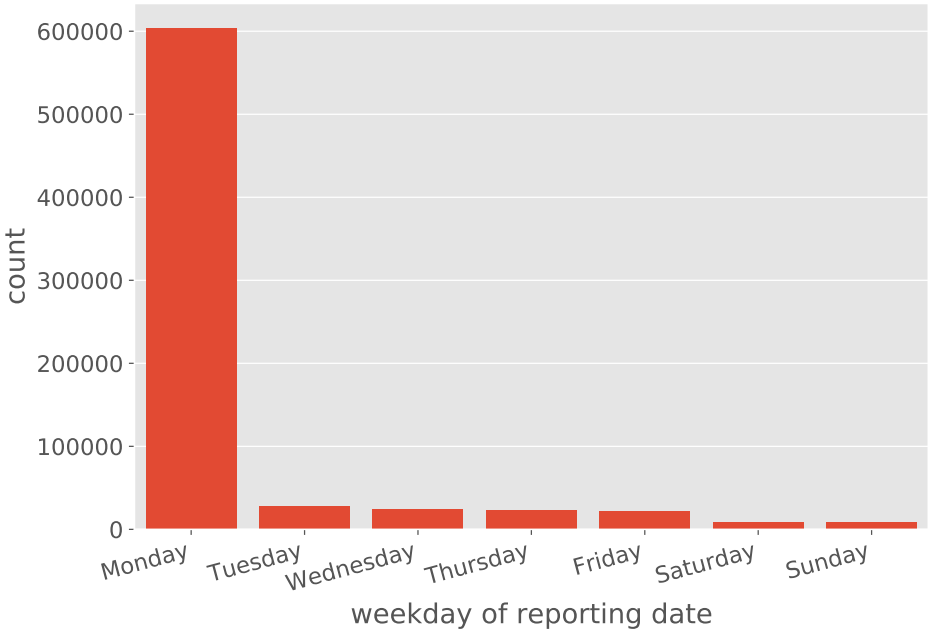


Figure 3.6: Distribution of weekdays of reporting since 2001.

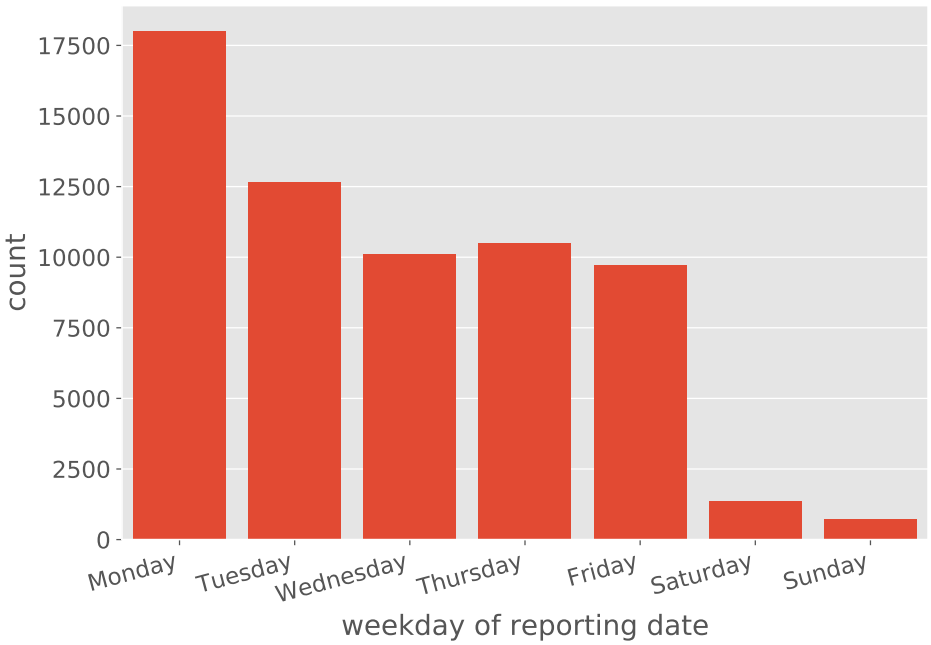


Figure 3.7: Distribution of weekdays of reporting since 2015.

3.2 Construction of the dataset

For the reasons described in Section 3.1 case records were aggregated weekly by county and by pathogen subtype for the two diseases *Salmonella* and *Campylobacter* respectively. To not run algorithms on many time series containing more or less no data at all we filtered all resulting time series in their final state by the following criteria:

- each time series needs to have its first case at least in the year 2004
- each time series needs to have its last case not earlier than in they year 2018
- each time series needs to contain at least two outbreaks between the beginning of 2016 and the end of 2017
- each time series needs to have on average one case per 30 days

After filtering by those criteria 137 time series were left for *Salmonella* and 169 for *Campylobacter*. All time series were sorted alphabetically according to county and pathogen subtype. For convenience reasons only the first 137 time series were kept for *Campylobacter*.

When an outbreak detection algorithm was applied to a times series corresponding to a filter combination, the time series was used in the following way: to get a prediction for a certain week t the time series from 2004 to $t-1$ using the data state from week t was used to fit the model to the data. Past cases known to be outbreak cases at time t where removed from the weekly case counts, so the algorithms could only fit the non-epidemic state. Then the data point at week t was used to predict an outbreak. This way the time series grew one week longer for each consecutive prediction.

The time between the beginning of 2016 and the end of 2017 was used for the hyperparameter optimization. The time in 2018 were used for evaluation.

3.3 Optimization

BOHB was run with minimum budget $b_{\min} = 15$, maximum budget $b_{\max} = 137$ and budget progression parameter $\eta = 3$ using five workers in parallel. Each optimization was run for 100 iterations. A separate optimization was run for each algorithm on each of the two diseases. Budgeting was implemented by only evaluating on the first b time series, with b being the current budget. Because BOHB expects an evaluation

function that can be minimized we replaced the score defined in Section 2.5 with a loss $l(\mathbf{o}, \mathbf{a}) = 1 - s(\mathbf{o}, \mathbf{a})$. We used the mean of the losses for all time series used at budget b as the final evaluation function.

For each outbreak detection algorithm we made the space of possible configurations very broad. In cases where values had a clearly defined range, we used the full range. For example we allowed the α threshold values to vary between 0 and 1. For values where the range of possible values was not constrained, we allowed broad variation around the default parameters. We used uniform priors for all hyperparameters when randomly sampled by BOHB. The exact configuration spaces for all algorithms are summarized in Table 3.1 and Table 3.2.

To verify that the optimization worked as intended we looked at correlation across different budgets. BOHB can only work if performance on lower budgets is indeed indicative of performance on higher budgets i. e. losses should correlate across budgets. As the authors of BOHB note[1] correlations across budget should not be below 0.2. As Figure 3.8 shows correlations are much higher, so BOHB was suitable for our problem.

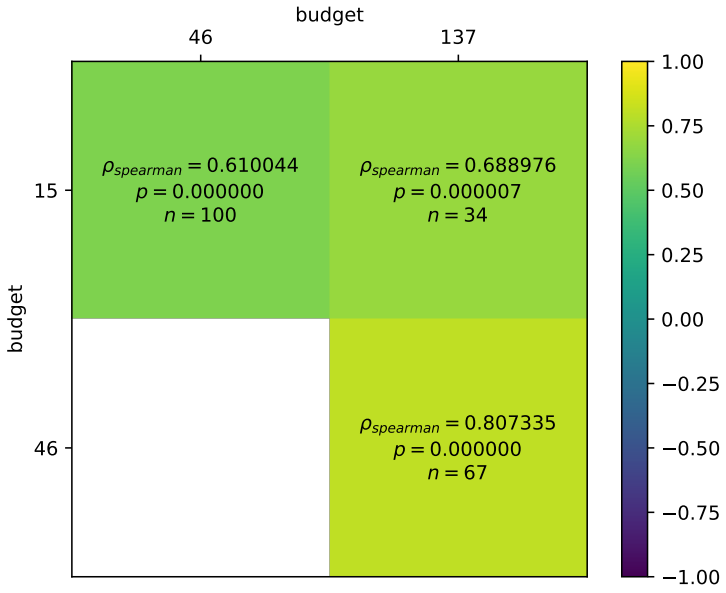


Figure 3.8: Correlations across different budgets for a BOHB run with the FarringtonFlexible algorithm on *Salmonella*.

[1] `https://automl.github.io/HpBandSter/build/html/best_practices.html`

Moreover, as Figure 3.9 shows, the losses do in fact decrease on average over time and the TPE model learns to produce good configurations as Figure 3.10 shows.

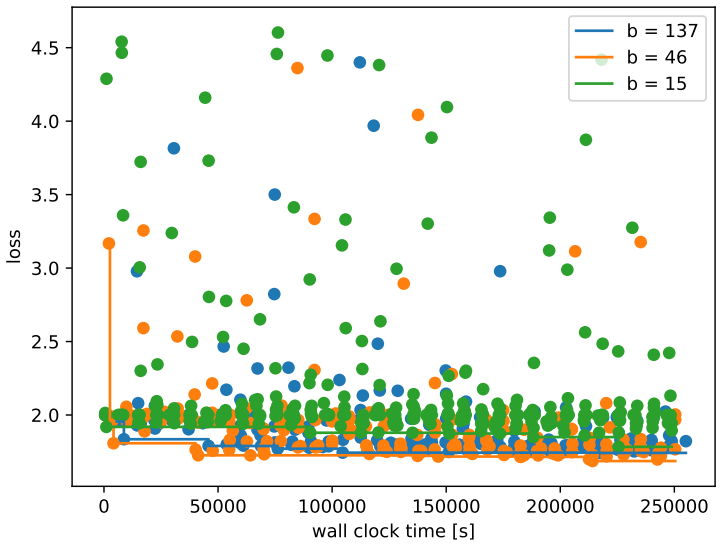


Figure 3.9: Losses over time for a BOHB run with the FarringtonFlexible algorithm on *Salmonella*. The lines track the minimum loss observed on each budget.

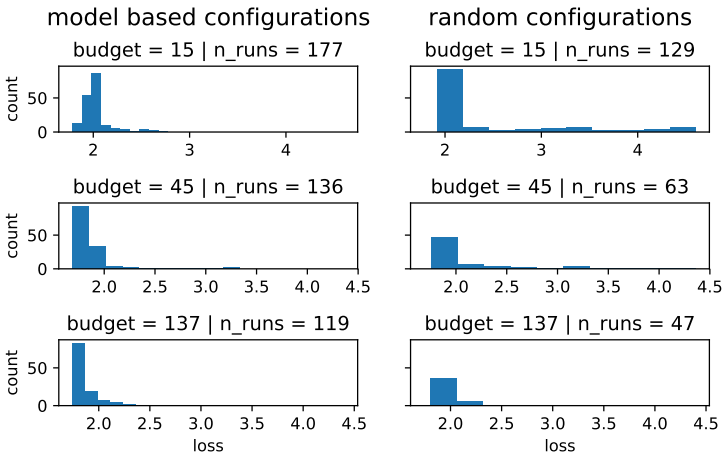


Figure 3.10: Losses for random configurations and model based configurations across different budgets for a BOHB run with the FarringtonFlexible algorithm on *Salmonella*.

Table 3.1: Configuration spaces for hyperparameter optimization.

algorithm	hyperparameter	value space
EarsC{1, 2}	alpha	$[0, 1]$
	baseline	$\{3, \ldots, 21\}$
	min_sigma	$[0, 3]$
CDC	years_back	$\{1, \ldots, 6\}$
	window_half_size	$1, 10$
	alpha	$[0, 1]$
RKI	years_back	$\{1, \ldots, 6\}$
	window_half_size	$\{1, \ldots, 6\}$
	include_recent_year	$\{T, F\}$
Farrington	years_back	$\{1, \ldots, 6\}$
	window_half_size	$\{1, \ldots, 10\}$
	reweight	$\{T, F\}$
	alpha	$[0, 1]$
	trend	$\{T, F\}$
	past_period_cutoff	$\{1, \ldots, 20\}$
	min_cases_in_past_periods	$\{1, \ldots, 20\}$
	power_transform	{2/3, 1/2, none}
FarringtonFlexible	years_back	$\{1, \ldots, 6\}$
	window_half_size	$\{1, \ldots, 10\}$
	reweight	$\{T, F\}$
	weights_threshold	$[0, 5]$
	alpha	$[0, 1]$
	trend	$\{T, F\}$
	trend_threshold	$[0, 2]$
	past_period_cutoff	$\{1, \ldots, 20\}$
	min_cases_in_past_periods	$\{0, \ldots, 20\}$
	power_transform	{2/3, 1/2, none}
	past_weeks_not_included	$\{1, \ldots, 52\}$
	threshold_method	{delta, nbPlugin, muan}

Table 3.2: Configuration spaces for hyperparameter optimization (continued).

algorithm	hyperparameter	value space
Cusum	reference_value	$[0, 3]$
	decision_boundary	$[0, 10]$
	expected_numbers_method	{glm, none}
	transform	{rossi, standard, anscombe, anscombe2nd, pearsonNegBin, anscombeNegBin, none}
	nbin_alpha	$[0, 3]$
GLRPoisson	glr_test_threshold	$\{0, \ldots, 10\}$
	change	{intercept, epi}
	direction	{inc, dec, {inc, dec}}
	upperbound_statistic	{cases, value}
GLRNegativeBinomial	alpha	$[0, 3]$
	glr_test_threshold	$\{0, \ldots, 10\}$
	change	{intercept, epi}
	direction	{inc, dec, {inc, dec}}
	upperbound_statistic	{cases, value}
	x_max	$[0, 10^5]$
OutbreakP	threshold	$\{0, \ldots, 200\}$
	upperbound_statistic	{cases, value}
	max_upperbound_cases	$\{10^3, \ldots, 10^6\}$
Bayes	years_back	$\{1, \ldots, 6\}$
	window_half_size	$\{1, \ldots, 10\}$
	include_recent_year	$\{T, F\}$
	alpha	$[0, 1]$

3.4 Performance

In this section we describe the influence of the hyperparameter optimization on the performance of the different algorithms. We compare the performance of the optimized configuration to the performance of the default configuration that are recommended by the R surveillance package. As a representative for all algorithm we look at the optimized scores and the overfitting for FarringtonFlexible in more detail.

As described in Section 3.3 we used the loss derived from the score developed in Section 2.5 as an optimization criterion. In the following, we mainly describe algorithm performance in terms of the score itself. In the graphics the score is denoted as "ghozzi score" in honor of its original proponent. In addition to the 'ghozzi score" we collected several other binary classification metrics [39].

To get an estimate for the default configuration's performance as a baseline and the potential for optimization, we compared the default configuration's mean score to the top 10% of the optimized configurations. Figure 3.11 and Figure 3.12 exemplify this comparison for FarringtonFlexible. The best optimized configurations' performances show indeed improvement over the default configuration. However, *Campylobacter* seems to pose a more difficult problem, as neither the default nor the optimized performance is nearly as strong as for *Salmonella*. The scores for *Salmonella* show that using the outbreak detection algorithm is better than never sounding an alarm, which would achieve a score of -1, both for the default and the optimized configurations. For *Campylobacter* the default configuration in fact achieves a worse score than the policy of never sounding an alarm would achieve (cf. Section 2.5). Optimization yields configurations that are slightly above that baseline.

Figure 3.11: Scores of the top-10% optimized configurations evaluated on full budget for FarringtonFlexible on *Salmonella* compared to the default configuration.

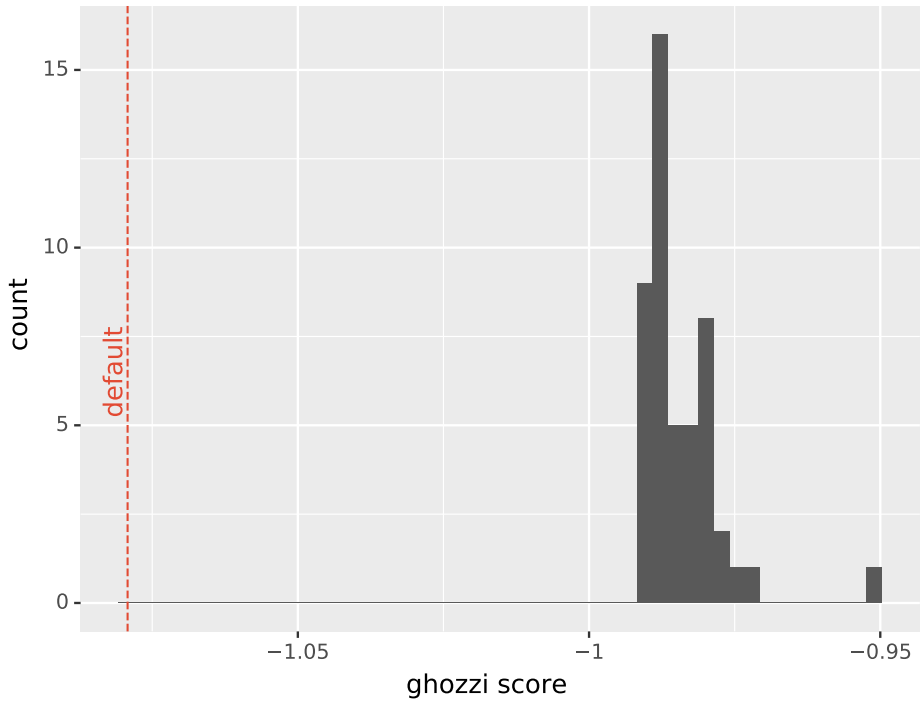


Figure 3.12: Scores of the top-10% optimized configurations evaluated on full budget for FarringtonFlexible on *Campylobacter* compared to the default configuration.

We used the best score achieved on the training set to provide a ranking for the different algorithms. For *Salmonella* we find that Farrington, Ears and Bayes perform best. Cusum-based algorithms seem to perform worse, and the RKI and CDC algorithms are last. Because the performance of OutbreakP is much worse for both *Salmonella* and *Campylobacter* it is not considered in the further analyses.

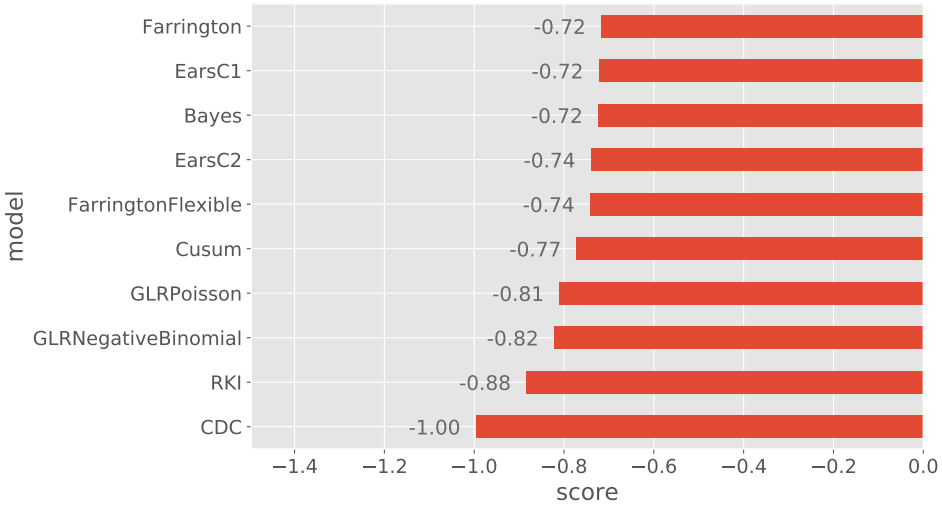


Figure 3.13: Ranking of algorithms for *Salmonella* based on the training set.

For *Campylobacter* the picture looks different. Farrington is still second best, but the distance between GLM-based/Ears and Cusum-based approaches has vanished. Bayes performs much worse compared to its performance on *Salmonella*. CDC still shows the weakest performance.

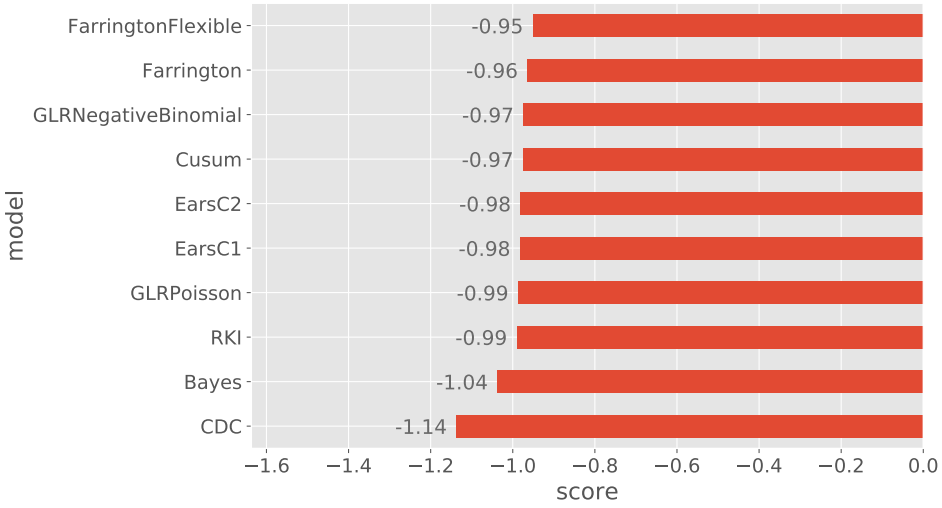


Figure 3.14: Ranking of algorithms for *Campylobacter* based on the training set.

Next we quantify the overfitting induced by the hyperparameter optimization. A problem with hyperparameter optimization can be that, as now the hyperparameters are learned as well, they overfit on the data the hyperparameter optimization used

for training. To estimate the overfitting we took the 10% best configurations on the training set and evaluated them on the test set. We plot kernel density estimates of the distribution of the means of all metrics for training and test set. The degree by which the training distribution contains higher values than the test distributions is an indicator of overfitting. While we show the distribution of all metrics here, only the distribution of the optimized criterion ("ghozzi score") determines whether there is overfitting or not. For *Salmonella* there seems to be little overfitting as the distributions for the training and test set largely overlap (Figure 3.15). For *Campylobacter* indeed all configurations seem to have overfitted the training data, even though the distributions are not extremely far apart (Figure 3.16).

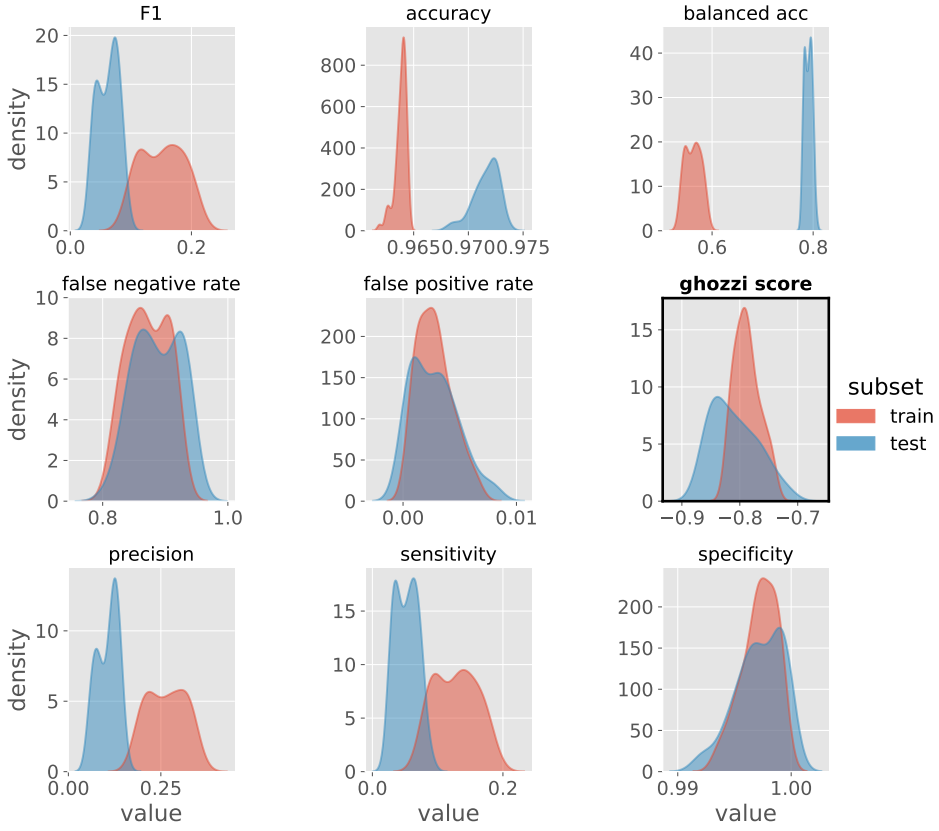


Figure 3.15: Kernel density estimate of distribution of different metrics on training and test set for top-10% configurations on the training set for FarringtonFlexible on *Salmonella*. The optimization criterion is highlighted.

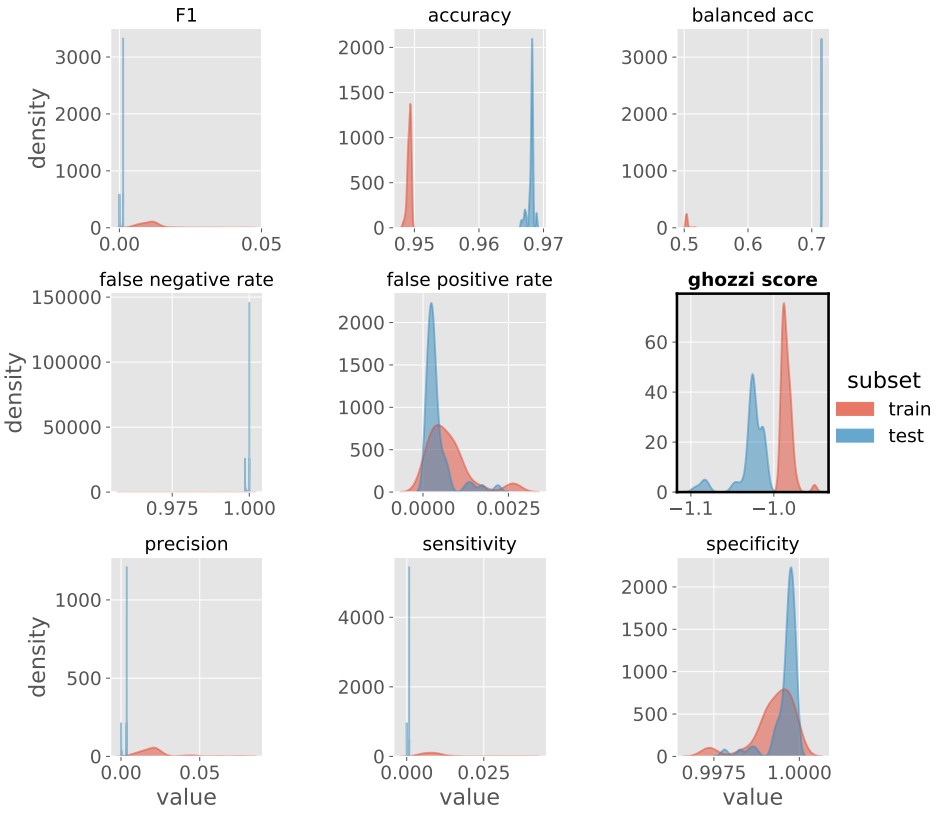


Figure 3.16: Kernel density estimate of distribution of different metrics on training and test set for top-10% configurations on the training set for FarringtonFlexible on *Campylobacter*. The optimization criterion is highlighted.

Finally, we look at the performance of the best configurations (selected based on the performance on the training set) on the test set. In Figure 3.17 and Figure 3.18 we show the difference in mean validation score between the default configuration and the best configuration on the training set for each algorithm. We see that the improvements over the default configuration generalize to the test set in almost all cases. The only exception is Farrington on *Campylobacter* which got a bit worse.

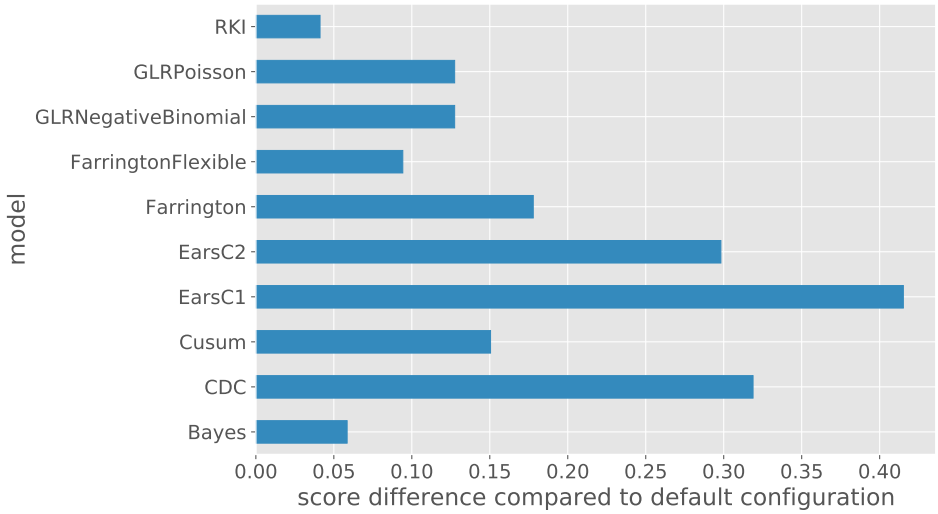


Figure 3.17: Improvement of the best configuration over the default configuration for *Salmonella*. The best configuration was picked based on the performance on the training set. The comparison with the default configuration is based on the test set. Improvements are shown in blue, deteriorations in red.

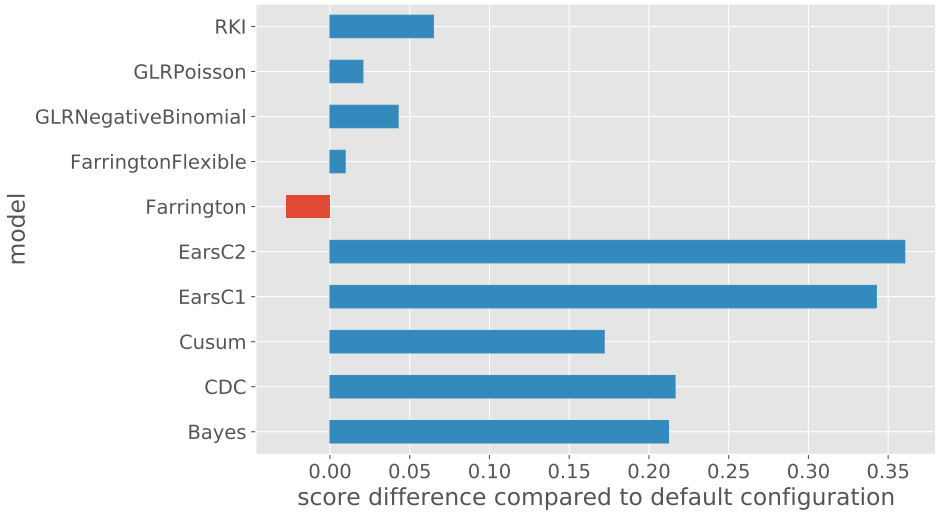


Figure 3.18: Improvement of the best configuration over the default configuration for *Campylobacter*. The best configuration was picked based on the performance on the training set. The comparison with the default configuration is based on the test set. Improvements are shown in blue, deteriorations in red.

The same result is presented in Figure 3.19 and Figure 3.20 in a slightly different way. As a separate score is produced for each of the 137 filter combinations used, we in fact get a distribution of scores that we usually summarize with the mean. In Figure 3.19 and Figure 3.20 we show the distribution of scores of the best optimized and default configuration on the test set for each algorithm. Again we see that optimization yielded notifiable improvements for almost all algorithms.

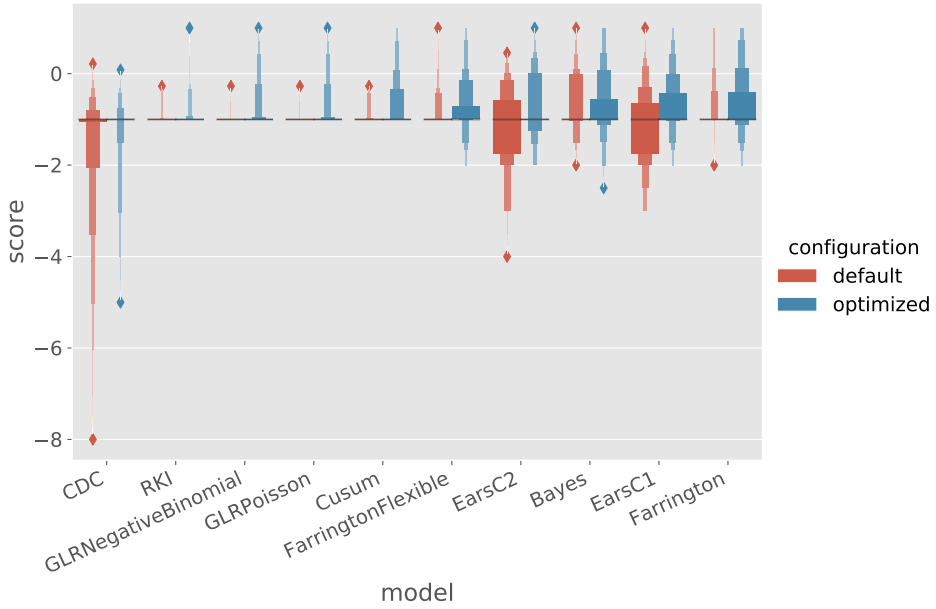


Figure 3.19: Comparison of the score distribution for the best configuration and the default configuration on *Salmonella*. Metrics are calculated on the test set.

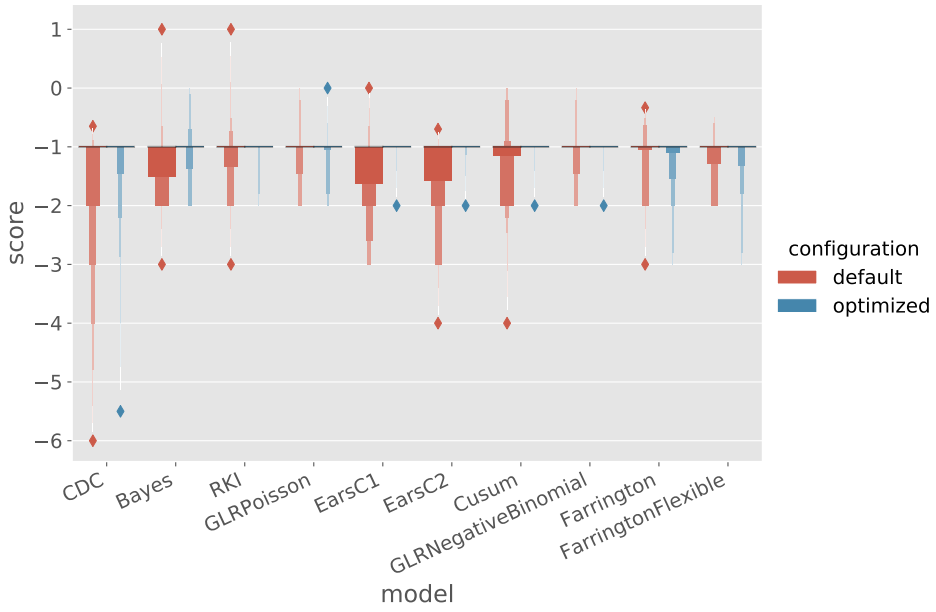
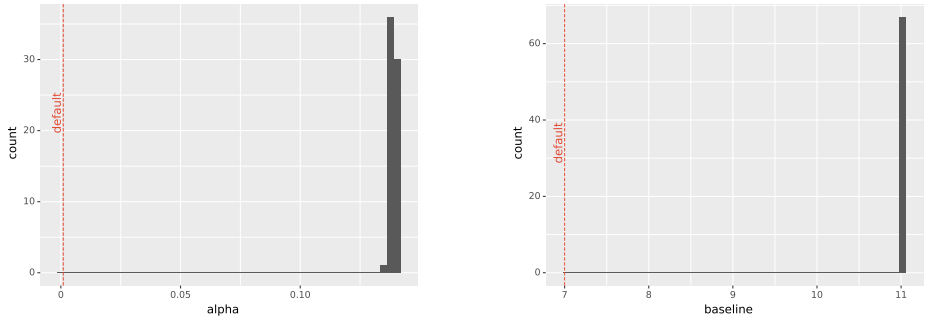


Figure 3.20: Comparison of the score distribution for the best configuration and the default configuration on *Campylobacter*. Metrics are calculated on the test set.

3.5 Interpretation of Optimized Hyperparameters

In this section we exemplary look at the behavior of optimized parameters for EarsC1 and Farrington for *Salmonella*. We chose these algorithms as examples, because EarsC1 benefitted the most from optimization and Farrington performed best on *Salmonella* (Figure 3.17).

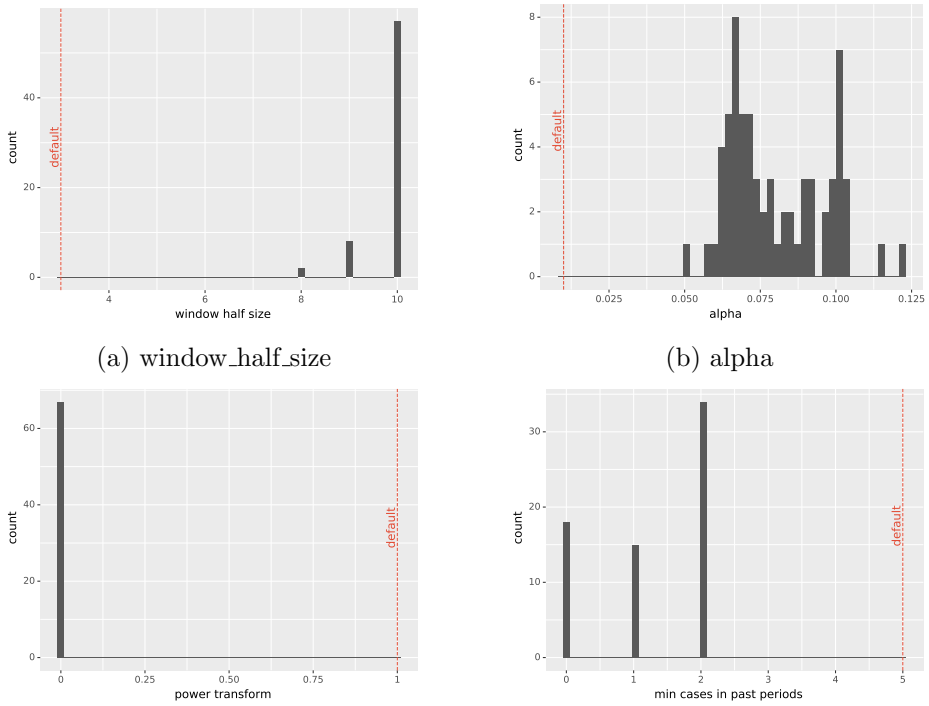
For EarsC1 there are two parameters of interest: *alpha*, that determines the alerting threshold, and *baseline*, that determines how many past weeks are used to form the estimate of the current week. We see in Figure 3.21a that the alerting threshold was lowered and in Figure 3.21b we see that more weeks were taken into account for optimized configurations. The *min_sigma* parameter is not considered here as it only takes effect when $baseline = 0$, which never is the case.



(a) alpha (b) baseline

Figure 3.21: Parameter distributions of the top-10% EarsC1 configurations on *Salmonella*.

For Farrington we find that the parameters $window\_half\_width$, $alpha$, $power\_transform$ and $min\_cases\_in\_past\_periods$ differ significantly from the default configuration. $window\_half\_width$ which determines how many weeks around the corresponding point in previous years are taken into account to estimate seasonality is increased (Figure 3.22a). This signifies that it makes sense to use more information to estimate seasonality. The *alpha* parameter is increased as well, which should correspond to an increase in sensitivity (Figure 3.22b). All optimized models use the variance stabilizing power transform instead of the default skewness correction power transform (Figure 3.22c). Finally, $min\_cases\_in\_past\_periods$ is lowered. This parameter is used for alarm suppression. If the number of cases in the past few weeks does not exceed $min\_cases\_in\_past\_periods$ then no alarm is raised. As the time series for *Salmonella* are very sparse and characterized by low case numbers, it only makes sense that this threshold is lowered as otherwise no alarm could ever be raised (Figure 3.22d).



(a) window_half_size

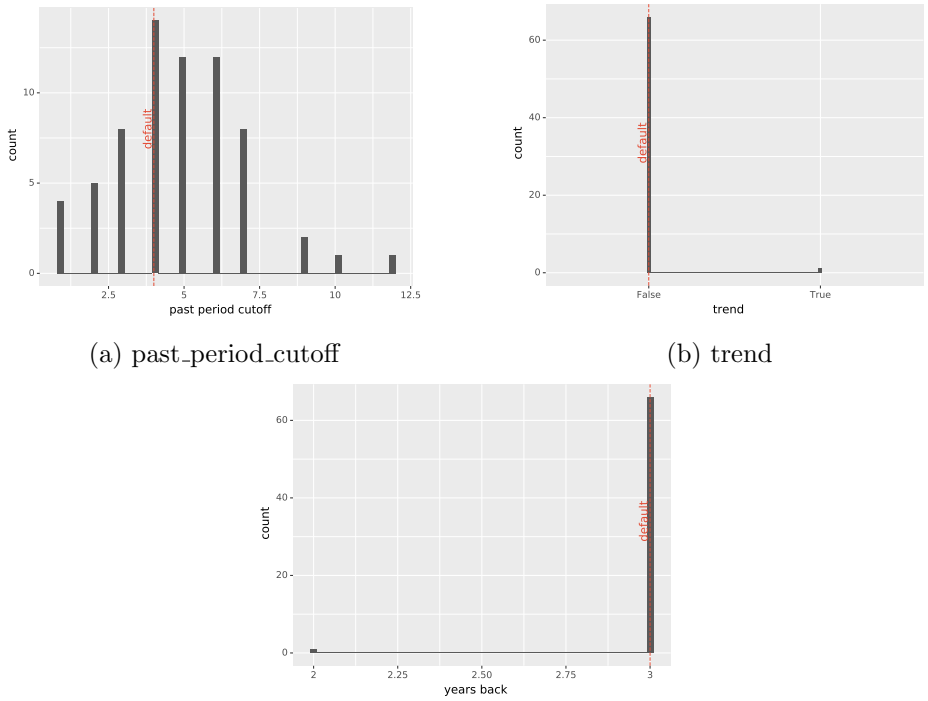
(b) alpha

(c) power_transform

(d) min_cases_in_past_periods

Figure 3.22: Parameter distributions for the top-10% Farrington configurations on *Salmonella* that differ from the default configuration.

The parameters $past\_period\_cutoff$, trend and $years\_back$ are very similar to the default configuration. Like the default configuration almost all optimized configuration do not include a trend (Figure 3.23b) and to only fit the last three years, ignoring the older data (Figure 3.23c). The values for $past\_period\_cutoff$, i. e. how many weeks are used to obtain the case count that has to exceed $min\_cases\_in\_past\_periods$, are distributed around the default value of four.



(a) past_period_cutoff

(b) trend

(c) years_back

Figure 3.23: Parameter distributions for the top-10% Farrington configurations on *Salmonella* that are similar to the default configuration.

Figure 3.24 summarizes the difference between default and optimized parameters in a tSNE embedding [40] in a two-dimensional space. We see that the default configuration is really on the edge of the subspace that the hyperparameter optimization found to contain good configurations.

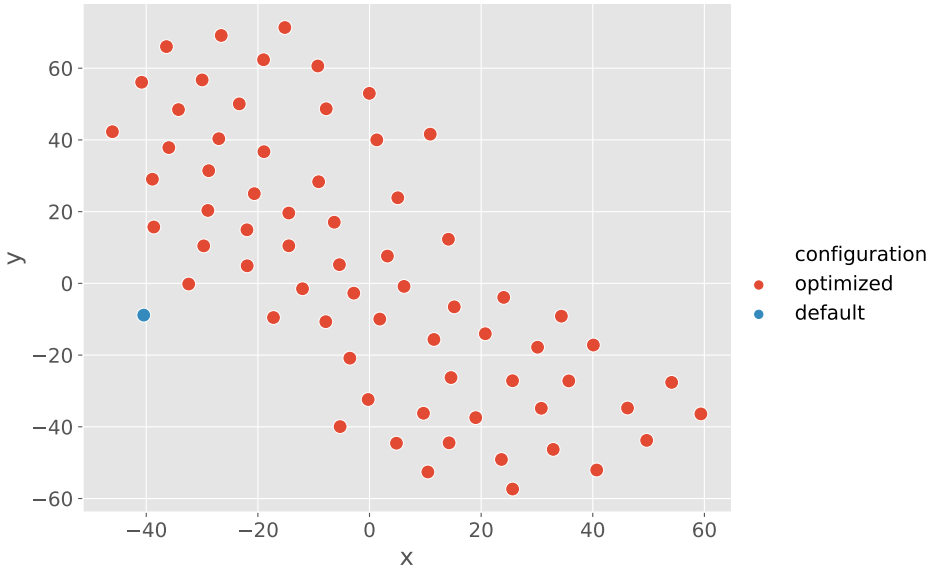


Figure 3.24: tSNE embedding of parameter distributions of the top-10% Farrington configurations on *Salmonella*.

3.6 Comparison to Baselines

To compare the outbreak detection algorithms to a simple baseline we used the fact that a lot of outbreaks are already known in the week they are to be detected (cf. Section 3.1.2). To find the best baseline we evaluated three different approaches: predicting an outbreak if there is an outbreak in the current week, predicting an outbreak if there is an outbreak in the past week and predicting an outbreak if there is an outbreak in the current *or* the last week. We found that the best performance is attained by only predicting outbreaks based on the current week for both *Salmonella* and *Campylobacter*. We subsequently call this method the baseline. For *Salmonella* the baseline achieves a score of -0.77 on the training set and -0.71 on the test set. In comparison, the configuration selected by the hyperparameter optimization, i. e. the best configuration for Farrington, achieves a score of -0.72 on the training set and -0.73 on the test set. The performance of the baseline is very comparable and even better on the test set.

For *Campylobacter* the baseline achieves a score of -0.66 on the training set and -0.69 on the test set. In comparison, the configuration selected by the hyperparameter optimization, here the best configuration for FarringtonFlexible, only achieves a score of -0.95 on the training set and -1.08 on the test set.

3.7 Optimization and Interpretation of Different Optimization Criteria

As the "ghozzi score" was developed based on assumptions about what would be desireable properties of outbreak detection algorithms, it is hard to estimate what exact influence it has on the behavior of outbreak detection algorithms when used as an optimization criterion. We therefore recorded other common metrics for binary classification alongside the "ghozzi score". Figure 3.25 shows the development of all these metrics over the course of the hyperparameter optimization. Each step represents another model that was evaluated on the full budget. All curves were smoothed using Loess smoothing [41]. The main interpretation of these curves is that the "ghozzi score" constrains configurations to be highly specific. Sensitivity can only increase up to around 0.1, while specificity stays at around 0.99. This indicates that the punishment for false positives might be way to strong.

Figure 3.25: Development of different metrics across the course of optimization for optimizing the "ghozzi score" score for FarringtonFlexible on *Salmonella*. The optimization criterion is highlighted.

The same conclusion can be drawn from Figure 3.26, which shows the correlation of the mean metrics from all configurations evaluated on the full budget. Again we see that the "ghozzi score" positively correlates with specificity, but negatively correlates with sensitivity.

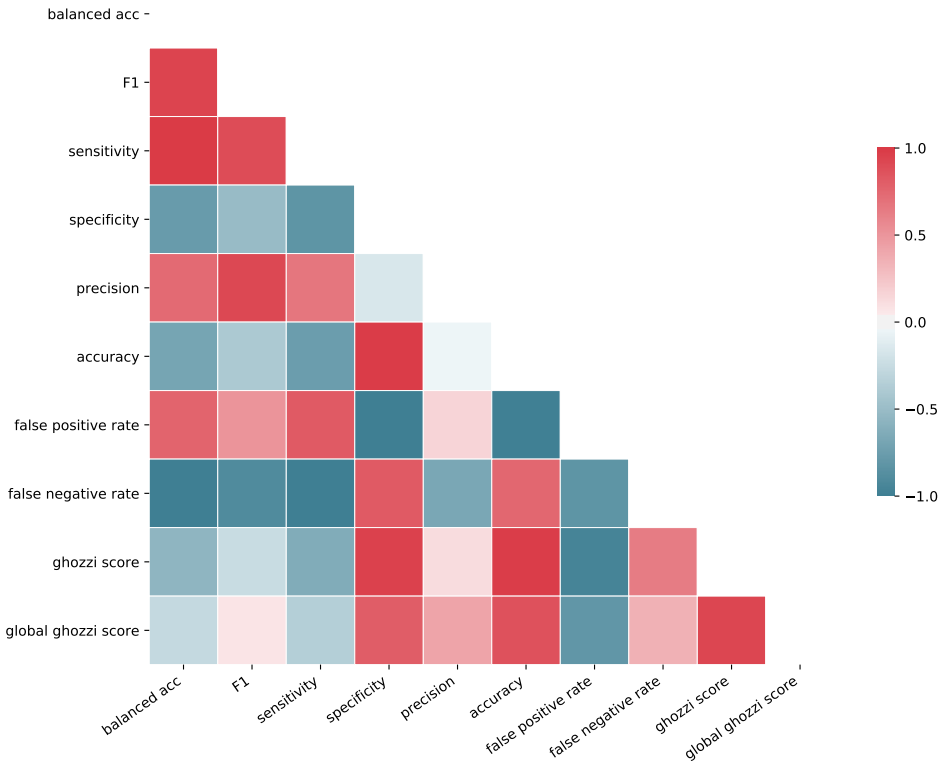


Figure 3.26: Correlation matrix between metrics for FarringtonFlexible on *Salmonella*.

Experimentally, we tried to optimize other criteria than the "ghozzi score". A very simple choice was to optimize the F1 score. Figure 3.27 shows the development of different binary metrics for optimizing the F1 score. It can be seen that sensitivity is strongly increased compared to the optimization based on the "ghozzi score" and reaches values up to 0.34. Specificity is slightly reduced to around 0.96.

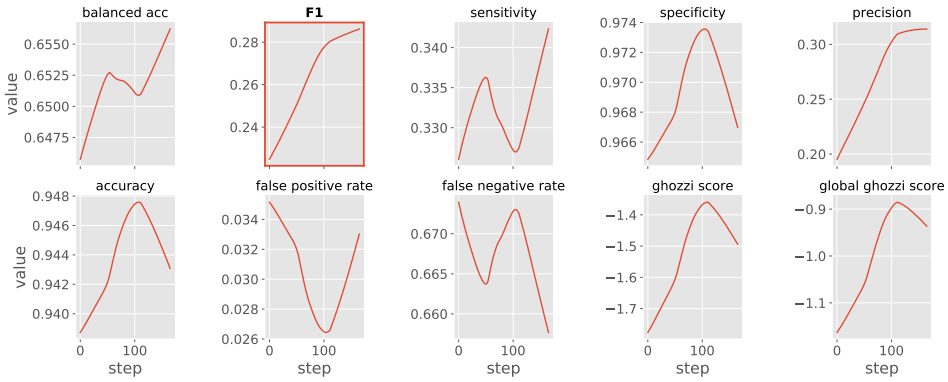


Figure 3.27: Development of different metrics across the course of optimization for optimizing the F1 score. The optimization criterion is highlighted.

Finally, we optimized a weighted F1 score, where each sample was weighted by the number of cases in the respective week (Figure 3.28). The results of optimizing a case weighted F1 score go in the same direction as the results of optimization the plain F1 score, but are even more pronounced. Sensitivity goes up to 0.5 and specificity decreases to 0.92.

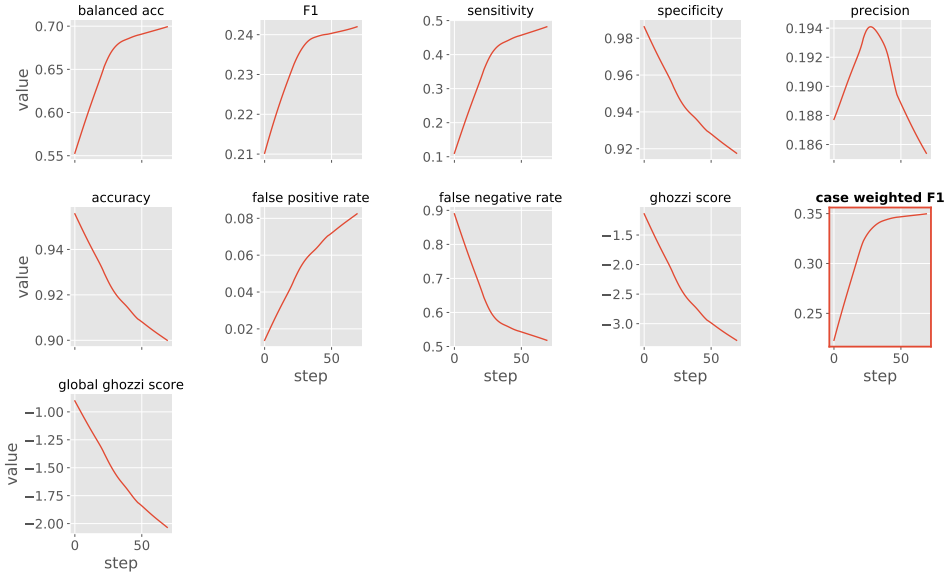


Figure 3.28: Development of different metrics across the course of optimization for optimizing the case weighted F1 score. The optimization criterion is highlighted.

While these results sound promising, we found after visual inspection that optimizing F1 scores leads to configurations that almost always predict outbreaks, when there are at least a few cases in the respective week. The high numbers for specificity are misleading, because the metric is calculated over all weeks, but many weeks contain no cases. Weeks without cases are easy to classify correctly for every outbreak detection algorithm.

4 Conclusion

This study demonstrates how a framework for a fair comparison of outbreak detection algorithms should look like. While all the parts are in place, all components require more consideration and fine tuning.

We showed that it is possible to apply hyperparameter optimization to outbreak detection algorithms and reliably obtain generalizable improvement. However, we found that no optimized configuration is better than a simple baseline, that utilizes information about outbreaks available in the current week. Given these findings, we should rethink what information is communicated to the consumers of the generated signals, i. e. epidemiologist and others responsible for public health, and how we optimize our models towards the needs of those consumers. It should be considered whether a signal should always be raised if there are outbreak cases present in the current week. If that is not desired we need to redefine what exactly constitutes a label for a desired signal. One possibility would be to only do the labeling based on outbreaks that at least include a certain number of cases. This could make sense as larger outbreaks pose a stronger threat to public health as small outbreaks that only contain a few cases. In general, it would make sense to not only label epidemiological outbreaks, but also public health risks directly by experts to improve the applicability of supervised learning methods. This could be incorporated by giving the consumers of the generated signals the ability to directly mark signals as useful.

We developed a score that summarizes outbreak detection performance in a single number. However, optimizing this score leads to configurations that show very low sensitivity. This could be rooted in the fact that the score is unbounded towards negative infinity, so that punishment for to many false positives add up too strongly. Going back to a probabilistic interpretation, one could consider utilizing variants of proper scoring rules [42] as was done by Enki *et al.* [5]. Another interesting direction is cost-sensitive learning [43, 44]. In cost-sensitive learning different misclassification can have different cost, depending on the type of misclassification. Additionally, other types of cost can be incorporated. While weighting missing an outbreak dif-

ferently than falsely predicting an outbreak was already done in the "ghozzi score", it was done without regard to the theory behind cost-sensitive learning. Moreover, external cost factors, such as whether the burden of disease surpasses the public health capabilities could be incorporated in a consistent fashion.

A comparison with existing evaluation studies is difficult because of the different metrics, parameters and data used. However, some common binary classification metrics can still be compared. The study that comes closest to our approach is probably Bédubourg & Le Strat [6], as they also compare outbreak detection algorithms from the surveillance package, but using simulated instead of real data. Bédubourg & Le Strat [6] find sensitivity of evaluated algorithms to vary between 0.2 and 0.8 and specificity to vary mostly between 0.8 and 1. On our real data we found specificity to be always high, but sensitivity to be much lower, even for the default configurations. Enki *et al.* [5] follow a similar approach as Bédubourg & Le Strat [6]. While Enki *et al.* [5] start with real data, their identification of outbreaks is ultimately based on assumptions about the statistical processes that govern epidemics and not on expert labels. Amongst other metrics they measure false positive rate and find it to vary from 0.01 for time series with little background up to 0.25 for time series with many background cases. In our results the false positive rate always stays very low irregardless of the background. This comparison suggest that the performance of outbreak detection algorithms on real expert labeled data can not easily be inferred from simulation studies.

The relatively weak performance of well established outbreak detection algorithms on real data suggests that the approach of detecting outbreaks on univariate time series of case counts is too limited. The information contained in univariate time series might simply be not sufficient to recover expert labeled outbreaks. As discussed in Section 1.3 epidemiologists work on single case records that contain complex features. To close the gap between experts' practice and automated surveillance, the design of outbreak detection algorithms that operate on the same kind of data should be explored. The formalization in Section 1.3 is a first step in this direction. While case based outbreak detection has not been widely used in the surveillance community, the use of multivariate time series has long been advocated [19]. Models based on multivariate time series can be easy extensions of established techniques, while allowing for the incorporation of additional information, such as patient features or data from event-based surveillance, such as Google Trends [45]. Future work should explore both directions to overcome the limitation of the current approaches.

Acknowledgements

While this thesis mostly speaks of “we”, this being a thesis and “we” being mostly “I”, I want to thank the persons without whom this work would have hardly been possible. Firstly, I want to thank the Signale team for providing a welcoming atmosphere and many interesting games of table soccer. I want to especially thank Stéphane Ghozzi and Alexander Ullrich for supervising my thesis and, together with Auss Abbood, for many insightful discussions. In this respect I also want to thank Auss Abbood for many fun discussions during the time in the office. Finally, I want to express particular gratefulness to Conny and Hans for housing me during my time in Berlin.

Glossary

EI Epidemic Intelligence

EWAR Early Warning and Response

GLMs Generalized Linear Models

IBS Indicator-Based Surveillance

RKI Robert Koch Institute

TPE Tree-structured Parzen Estimators

WHO World Health Organization

5 Bibliography

1. World Health Organization. *Implementation of Early Warning and Response with a focus on Event-Based Surveillance* tech. rep. (2014), 1–64.
2. Merianos, A. & Peiris, M. *International Health Regulations (2005)* `https://apps.who.int/iris/bitstream/handle/10665/43883/9789241580410%7B%5C_%7Deng.pdf;jsessionid=9AE2446AA7CE0E7589193397F345914B?sequence=1`.
3. Niemer, U. Das neue Infektionsschutzgesetz (IfSG). *Das Gesundheitswesen* **63,** 136–138 (Aug. 2002).
4. Heudorf, U., Eikmann, T. & Exner, M. Rückblick auf 10 Jahre Infektionsschutzgesetz. *Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz* **56,** 455–465 (Mar. 2013).
5. Enki, D. G. *et al.* Comparison of statistical algorithms for the detection of infectious disease outbreaks in large multiple surveillance systems. *PLoS ONE* **11** (ed Schanzer, D. L.) e0160759 (Aug. 2016).
6. Bédubourg, G. & Le Strat, Y. Evaluation and comparison of statistical methods for early temporal detection of outbreaks: A simulation-based study. *PLoS ONE* **12** (ed Olson, D. R.) e0181227 (July 2017).
7. Jafarpour, N., Izadi, M., Precup, D. & Buckeridge, D. L. Quantifying the determinants of outbreak detection performance through simulation and machine learning. *Journal of Biomedical Informatics* **53,** 180–187 (Feb. 2015).
8. Wang, X. *et al.* Comparing early outbreak detection algorithms based on their optimized parameter values. *Journal of Biomedical Informatics* **43,** 97–103 (Feb. 2010).
9. World Health Organization. Communicable disease surveillance and response systems - Guide to monitoring and evaluating. *WHO,* 90 (2006).
10. Salmon, M. *et al.* A system for automated outbreak detection of communicable diseases in Germany. *Eurosurveillance* **21,** 1 (2016).

11. Salmon, M. *Advances in Count Time Series Monitoring for Public Health Surveillance* PhD thesis (Ludwig–Maximilians–Universität München, 2016), 134.

12. Noufaily, A. *et al.* An improved algorithm for outbreak detection in multiple surveillance systems. *Statistics in Medicine* **32,** 1206–1222 (2013).

13. Ghozzi, S. *RKI - Signale - Early Warning System* `https://www.rki.de/EN/Content/infections/epidemiology/signals/signals%7B%5C_%7Dnode.html` (2019).

14. World Health Organisation. *WHO — Disease outbreaks* `https://www.who.int/environmental%7B%5C_%7Dhealth%7B%5C_%7Demergencies/disease%7B%5C_%7Doutbreaks/en/` (2019).

15. Brady, O. J., Smith, D. L., Scott, T. W. & Hay, S. I. Dengue disease outbreak definitions are implicitly variable. *Epidemics* **11,** 92–102 (June 2015).

16. Australian Government Department of Health. *Department of Health — Chapter 6: Outbreaks and case definitions* `http://www.health.gov.au/internet/publications/publishing.nsf/Content/cda-cdna-norovirus.htm-l%7B~%7Dcda-cdna-norovirus.htm-l-6` (2019).

17. Ester, M., Hans-Peter, K., Jorg, S. & Xiaowei, X. Density-Based Clustering Algorithms for Discovering Clusters. *Comprehensive Chemometrics* **2,** 635–654 (2010).

18. Dietterich, T. G. *Machine Learning for Sequential Data: A Review* in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (Springer, Berlin, Heidelberg, 2002), 15–30. doi:`10.1007/3-540-70659-3_2`.

19. Unkel, S., Farrington, C., Garthwaite, P., Robetson, C. & Andrews, N. Statistical models for the detection of infectious disease outbreaks: a review. *J R Statist Soc A* **175,** 49–82 (2012).

20. Salmon, M., Schumacher, D. & Höhle, M. Monitoring Count Time Series in R : Aberration Detection in Public Health Surveillance. *Journal of Statistical Software* **70.** doi:`10.18637/jss.v070.i10` (2016).

21. Chandola, V., Banerjee, A. & Kumar, V. Anomaly detection. *ACM Computing Surveys* **41,** 1–58 (July 2009).

22. Hutwagner, L., Thompson, W., Seeman, G. M. & Treadwell, T. The bioterrorism preparedness and response Early Aberration Reporting System (EARS). *Journal of urban health : bulletin of the New York Academy of Medicine* **80,** i89–i96 (2003).

23. Stroup, D. F., Williamson, G. D., Herndon, J. L. & Karon, J. M. *DETECTION OF ABERRATIONS IN THE OCCURRENCE OF NOTIFIABLE DISEASES SURVEILLANCE DATA* tech. rep. (1989), 323–329.

24. Farrington, C. P., Andrews, N. J., Beale, A. D. & Catchpole, M. A. A Statistical Algorithm for the Early Detection of Outbreaks of Infectious Disease. *Journal of the Royal Statistical Society. Series A (Statistics in Society)* **159,** 547 (1996).

25. Oakland, J. S. *Statistical Process Control: Sixth Edition* 1–458. doi:10.4324/9780080551739 (Routledge, Sept. 2007).

26. Rossi, G., Lampugnani, L. & Marchi, M. an Approximate Cusum Procedure for. *Statistics in Medicine* **2122,** 2111–2122 (1999).

27. Höhle, M. *Höhle: Poisson regression charts for the monitoring of surveillance time series Projektpartner Poisson regression charts for the monitoring of surveillance time series* tech. rep. (2006).

28. Höhle, M. & Paul, M. Count data regression charts for the monitoring of surveillance time series. *Computational Statistics and Data Analysis* **52,** 4357–4368 (May 2008).

29. Frisén, M., Andersson, E. & Schiöler, L. Robust outbreak surveillance of epidemics in Sweden. *Statistics in Medicine* **28,** 476–493 (2009).

30. Höhle, M. Surveillance: An R package for the monitoring of infectious diseases. *Computational Statistics* **22,** 571–582 (2007).

31. Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. *Algorithms for Hyper-Parameter Optimization* in *Proceedings of Neural Information Processing Systems (NIPS), 2011* (2011), 1–9. doi:2012arXiv1206.2944S.

32. Falkner, S., Klein, A. & Hutter, F. BOHB: Robust and Efficient Hyperparameter Optimization at Scale (2018).

33. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. HYPERBAND : BANDIT - BASED CONFIGURATION EVALUATION FOR HYPERPARAMETER OPTIMIZATION. *ICLR,* 1–15 (2017).

34. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research* **18,** 1–52 (2016).

35. Buitinck, L. *et al.* API design for machine learning software: experiences from the scikit-learn project (Sept. 2013).

36. Mckinney, W. *Data Structures for Statistical Computing in Python* in *PROC. OF THE 9th PYTHON IN SCIENCE CONF* (2010), 51.

37. Taylor, S. J. & Letham, B. Forecasting at Scale. *American Statistician* **72,** 37–45 (Jan. 2018).

38. Greff, K., Klein, A., Chovanec, M., Hutter, F. & Schmidhuber, J. *The Sacred Infrastructure for Computational Research* in *Proceedings of the 16th Python in Science Conference* (SciPy, 2017), 49–56. doi:10.25080/shinma-7f4c6e7-008.

39. M, H. & M.N, S. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process* **5,** 01–11 (2015).

40. Van Der Maaten, L. J. P. & Hinton, G. E. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research* **9,** 2579–2605 (2008).

41. Jacoby, W. G. Loess: a nonparametric, graphical tool for depicting relationships between variables. *Electoral Studies* **19,** 577–613 (Dec. 2000).

42. Czado, C., Gneiting, T. & Held, L. Predictive model assessment for count data. *Biometrics* **65,** 1254–1261 (Dec. 2009).

43. Zhou, Z.-H. in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 17–18 (Springer US, Boston, MA, 2011). doi:10.1007/978-3-642-22589-5_2.

44. Elkan, C. *The foundations of cost-sensitive learning* in *IJCAI International Joint Conference on Artificial Intelligence* (2001), 973–978.

45. Pervaiz, F., Pervaiz, M., Rehman, N. A. & Saif, U. FluBreaks: Early epidemic detection from google flu trends. *Journal of Medical Internet Research* **14.** doi:10.2196/jmir.2102 (2012).