

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Preventing Multiple Comparisons Problems in Data Exploration and Machine Learning

Permalink

<https://escholarship.org/uc/item/6h32g578>

Author

Koulouris, Nikolaos

Publication Date

2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Preventing Multiple Comparisons Problems in Data Exploration and Machine Learning

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Nikolaos Koulouris

Committee in charge:

Professor Yannis Papakonstantinou, Chair
Professor Sanjoy Dasgupta
Professor Alin Deutsch
Professor Arun Kumar
Professor Kevin Patrick

2020

Copyright

Nikolaos Koulouris, 2020

All rights reserved.

The Dissertation of Nikolaos Koulouris is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

DEDICATION

To my family and friends.

EPIGRAPH

It is pointless to do with more
what can be done with fewer.

Occam

Ithaka gave you the marvelous journey.
Without her you would not have set out.
She has nothing left to give you now.

Constantinos P. Cavafy

Never give up control.
Live life on your own terms.

W. White

3.5.2	Interactive Data Exploration	35
3.6	Data-driven Hierarchies	35
3.7	Analysis of Gain in Power	37
3.8	Experimental Evaluation	39
3.8.1	Experimental Design	40
3.8.2	Statistical Power Gain Against Baselines	41
3.8.3	Real Public Health Data Set	42
3.8.4	Data driven Hierarchy	43
3.8.5	Isolating Effects on Power	44
3.9	Related Work	45
3.10	Conclusion	47
Chapter 4	Avoiding Validation Overfitting during Feature Selection	51
4.1	Motivating Example & Overview	51
4.2	Background	54
4.2.1	ML Concepts	54
4.2.2	ThresholdOut	57
4.3	Analysis of ThresholdOut	58
4.3.1	Design of Experimental Analysis	58
4.3.2	Forward Selection	59
4.3.3	Backward Selection	60
4.3.4	Guidelines for setting τ , σ	61
4.4	AUTO-SET	62
4.4.1	Forward Selection	64
4.4.2	Backward Selection	65
4.5	Auto-Adjust Threshold feature selection	66
4.5.1	AAT Overview	67
4.5.2	Approximating validation error curve	68
4.5.3	Forward Selection	70
4.5.4	Backward Selection	72
4.5.5	Wrapper methods	73
4.6	Experiments	74
4.6.1	Experimental Design	75
4.6.2	Analysis of ThresholdOut	77
4.6.3	AUTO-SET	78
4.6.4	Auto-Adjust Threshold	79
4.7	Related Work	80
4.8	Conclusion	82
Chapter 5	Conclusion	100
Bibliography	102

LIST OF FIGURES

Figure 1.1.	Paradigm shift from hypothesis-driven to data-driven studies.	2
Figure 1.2.	How data exploration systems create multiple hypotheses	3
Figure 1.3.	How autoML systems create multiple hypotheses	5
Figure 2.1.	Visualizing events and probabilities in hypothesis testing.	12
Figure 3.1.	Visualizing data exploration results of a public health data set.	21
Figure 3.2.	Small part of input hierarchy presented in Figure 3.1.	27
Figure 3.3.	Visualizing the correlation between child variables of two different group factors between which the Group Quality algorithm has to pick.	31
Figure 3.4.	Experiment of different data exploration algorithms for varying parameters of data with two different controlling techniques.	48
Figure 3.5.	Comparison of Statistical power of our algorithms with data-driven(DD) and a priori(A-p) hierarchies.	49
Figure 3.6.	Experiment with real Public Health data set.	49
Figure 3.7.	Experiment to show how the sample size and the effect of a factor on an outcome affect statistical power.	50
Figure 4.1.	Training, validation and test error during feature selection in four different scenarios.	52
Figure 4.2.	Summary of results of analysis of ThresholdOut algorithm in Forward and Backward Selection.	59
Figure 4.3.	Bias-variance trade-off.	64
Figure 4.4.	Overview of Auto Adjust Threshold feature selection process.	68
Figure 4.5.	Heatmap of test error (e_{res}) and overfitting (o_{res}) for forward and backward selection in the synthetic dataset.	83
Figure 4.6.	Heatmap of test error (e_{res}) and overfitting (o_{res}) for forward and backward selection in the heart dataset.	84
Figure 4.7.	Heatmap of test error (e_{res}) and overfitting (o_{res}) for forward and backward selection in the ion dataset.	85

Figure 4.8.	Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the synthetic dataset with Logistic Regression.	86
Figure 4.9.	Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the heart dataset with Naive Bayes.	87
Figure 4.10.	Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the cancer dataset with Logistic Regression.	88
Figure 4.11.	Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the sani dataset with Logistic Regression.	89
Figure 4.12.	Percentage of times $e_t < e_t^{b_1}$ in all 30 different settings.	90
Figure 4.13.	Percentage of times $o < o^{b_1}$ in all 30 different settings.	91
Figure 4.14.	Percentage of times $e_t < e_t^{b_2}$ in all 30 different settings.	92
Figure 4.15.	Percentage of times $o < o^{b_2}$ in all 30 different settings.	93

LIST OF TABLES

Table 2.1.	Possible outcomes in hypothesis testing.	11
Table 3.1.	Notation used in this paper.	26
Table 4.1.	Characteristics of the datasets used.	76
Table 4.2.	Notations of metrics.	77
Table 4.3.	Results of ThresholdOut analysis on synthetic data.	94
Table 4.4.	Results of ThresholdOut analysis on sani dataset.	95
Table 4.5.	Results of ThresholdOut analysis on heart dataset.	96
Table 4.6.	Results of ThresholdOut analysis on ion dataset.	97
Table 4.7.	Results of ThresholdOut analysis on cancer dataset.	98
Table 4.8.	Performance of AAT in different settings.	99

ACKNOWLEDGEMENTS

I am hugely grateful to my advisor Yannis Papakonstantinou for his support and guidance throughout my years as a graduate student. Always setting the bar high, he pushed me to improve my ideas and my ability to solve tough problems. He believed in my abilities from early on and let me do things independently, which taught me how to work and do research independently. I am also thankful for his support during the last year of my PhD, which was a difficult personal time in my life.

I am thankful to Arun Kumar for his mentorship and collaboration on research over the last years of my PhD. Our insightful discussions, his feedback and guidance helped me become a better researcher. Also, I would like to acknowledge and thank him for his open contribution to the diversity in the CSE department.

I am grateful to my committee members Alin Deutsch, Kevin Patrick and Sanjoy Dasgupta for their constructive comments and meaningful discussions.

I would like to thank the members of the CSE department and the DB lab at UCSD. They helped make this journey more enjoyable. Specifically, I would like to thank Yannis Katsis for being a great mentor during the first years of my PhD, when I most needed guidance. In addition, I would like to thank all the PhD students I had the pleasure of meeting, interacting and calling my friends at CSE: Costa, Vicky, Rana, Ainur, Vraj, Ben and Jamie. Navigating the process of getting a PhD together has been fun.

This journey would not be nearly as enjoyable without my friends. I am especially grateful to my best friends, Andrei and Reggie, who were always by my side, especially when I was going through a very difficult time. You will forever be part of my life. I am also grateful to John, Reaiah and Jacob for always having a great time together. I am grateful to Faidra and Kosti for always having stimulating conversations about grad school from the beginning which helped me push myself. Lastly, I am grateful to Panos for being the first one who believed in me and my dreams and helped me achieve them. I wouldn't be the person I am today without him.

Last but not least, I would like to thank my family. This journey wouldn't have been

possible without the support of my parents, Chrysoula and Alexandros, and my brother, Yannis. They have been supporting and believing in my dreams since day one. They were always there for me in any difficulty I faced and their support has been invaluable.

Work adapted in this Dissertation

Chapter 3 contains material adapted from the paper “VigilaDE: Avoiding False Discoveries with Hierarchical Data in Data Exploration Systems” by Nikolaos (Nikos) Koulouris; Arun Kumar; Yannis Papakonstantinou. It is currently under review for publication. The dissertation author was the primary investigator and author of this paper.

Chapter 4 contains material adapted from the paper “A practical feature selection algorithm to avoid validation overfitting in autoML systems” by Nikolaos (Nikos) Koulouris; Arun Kumar; Yannis Papakonstantinou. It is currently being prepared for publication. The dissertation author was the primary investigator and author of this paper.

VITA

- 2014-2015 Research Intern, Institute for the Management of Information Systems, Athens, Greece.
- 2015 Diploma in Electronic and Computer Engineering, National Technical University of Athens.
- 2019 Master of Science Degree in Computer Science, University of California San Diego.
- 2017 Software Development Engineer Intern, A9.Com, Amazon, Palo Alto, California.
- 2018 Software Development Engineer Intern, Redshift - AWS, Amazon, East Palo Alto, California.
- 2019 Software Development Engineer Intern, Redshift - AWS, Amazon, East Palo Alto, California.
- 2015-2020 Graduate Research Assistant, University of California San Diego
- 2020 Doctor of Philosophy in Computer Science, University of California San Diego

PUBLICATIONS

Nikos Koulouris, Arun Kumar, Yannis Papakonstantinou. “A practical feature selection algorithm to avoid validation overfitting in autoML systems”. *Under Submission*, 2020.

Nikos Koulouris, Arun Kumar, Yannis Papakonstantinou. “VigilaDE: Avoiding False Discoveries with Hierarchical Data in Data Exploration Systems”. *Under Review*, 2020.

Yannis Katsis, **Nikos Koulouris**, Yannis Papakonstantinou. “Assisting Discovery in Public Health”. *Human-In-the-Loop Data Analytics (HILDA), SIGMOD 2017*.

Yannis Katsis, Natasha Balac, Derek Chapman, Madhur Kapoor, Jessica Block, William G. Griswold, Jeannie Huang, **Nikos Koulouris**, Massimiliano Menarini, Viswanath Nandigam, Mandy Ngo, Kian Win Ong, Yannis Papakonstantinou, Besa Smith, Konstantinos (Costas) Zarifis, Steven Woolf, Kevin Patrick. “Big Data Discovery for Public Health”, *Chase 2017, Second IEEE/ACM Conference on Connected Health, Applications, Systems and Engineering Technologies*.

ABSTRACT OF THE DISSERTATION

Preventing Multiple Comparisons Problems in Data Exploration and Machine Learning

by

Nikolaos Koulouris

Doctor of Philosophy in Computer Science

University of California San Diego, 2020

Professor Yannis Papakonstantinou, Chair

More data means more opportunity for a researcher to test more hypotheses until he discovers an interesting finding. This increases the probability of arriving at a false conclusion purely by chance and is known as the multiple comparisons problem. Data exploration systems facilitate exploring big data by automatically testing thousands of hypotheses in order to find the most interesting ones. In machine learning analysts repeatedly test a model's performance on a holdout dataset until they find the one with the best performance. Auto-ML systems try to automate this model selection process. In both cases, testing for more things means a higher probability of making a statement purely by chance.

This dissertation examines how the multiple comparisons problem appears in the field

of data exploration and machine learning. In both cases we propose techniques that exploit some structure that appears in the field to improve upon existing techniques and reduce the consequences of multiple comparisons.

We present *VigilaDE*, the first data exploration system that utilizes the hierarchical structure of the data in order to control false discoveries. A plethora of real-world datasets already have domain-specific hierarchies that describe the relationship between variables. *VigilaDE* utilizes these hierarchies to guide the exploration towards interesting discoveries while controlling false discoveries and, as a result, increasing statistical power. Through extensive experiments with real-world data, simulations and theoretical analysis we show that our data exploration algorithms can find up to 2.7x more true discoveries in the data against the baseline while controlling the number of false discoveries.

In machine learning, testing multiple different models can lead to overfitting. We present an experimental analysis of *ThresholdOut*, the state of the art algorithm for avoiding overfitting to a holdout dataset in an adaptive setting. The main limitation of *ThresholdOut* is setting its parameters. We present *AUTO-SET*, an automated way to set its parameters specifically for feature selection. Specifically in feature selection the order of the models that we test on a holdout dataset has a very specific structure. We utilize this structure in *Auto Adjust Threshold*, a novel feature selection algorithm that avoids overfitting a holdout dataset and show that it outperforms existing algorithms.

Chapter 1

Introduction

1.1 Multiple Comparisons Problem

The recent explosion of data has created the opportunity to transition from hypothesis-driven [48] to data-driven discoveries and studies [63, 69]. In hypothesis-driven studies, researchers start by formulating a very specific hypothesis and then collect data to try and prove their initial hypothesis. In data-driven studies, researchers start by collecting as much data as possible and then they try to find any kind of relationship in the data that could be considered a new “discovery”. This paradigm shift can be seen in Figure 1.1. For example, one prominent professor congratulated a student for turning a dataset with no statistically important results into four published papers by testing multiple plausible hypotheses until some of them could be supported by the data [83]. One wonders how many of these hypotheses are true discoveries and not just spurious results based on randomness [81].

The multiple comparisons problem, also known as multiplicity or multiple hypotheses testing problem, appears very often in data-driven analyses where one considers multiple statistical hypotheses simultaneously [57]. The more hypotheses that one tests simultaneously, the higher the probability of *at least one* of them appearing to be correct due to randomness. As a result, users of data-driven analyses have to adhere to the correct statistical methods in order to avoid arriving to conclusions that appear to be true purely by chance. This is the reason why many data-driven studies can be questionable [45].

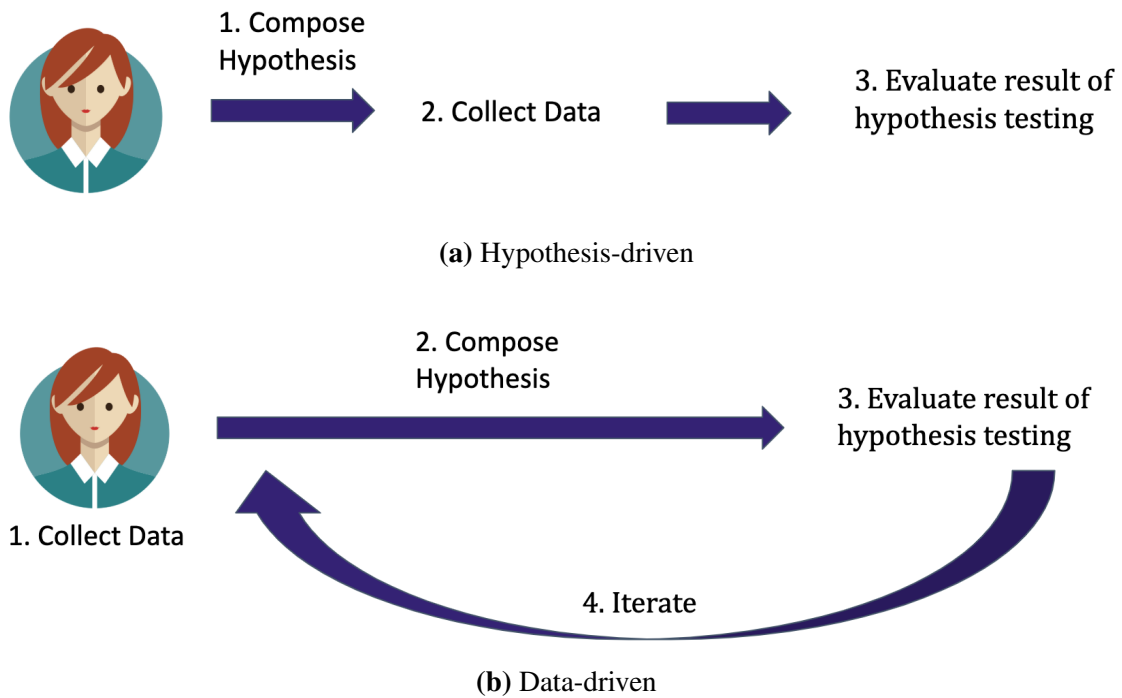


Figure 1.1. Paradigm shift from hypothesis-driven to data-driven studies.

In the medical field, all published research results have to report the statistical methods that were used to analyze the data [50]. This includes the number of hypotheses that were tested before arriving to a discovery and how they controlled for the problem of multiple hypothesis testing. Still, in the highly cited [45] Ioannidis observed that one of the key reasons why most research findings are false is the problem of multiple hypothesis testing and how scientists fail to adhere to acceptable ways of controlling random discoveries [59]. This problem has been identified in many other fields, such as ecology [89] and public health [47].

1.1.1 Multiple Comparisons Problem in Data Exploration Systems

New data exploration tools aim to automate and facilitate data exploration and analysis. Many recent works [49, 82] have proposed novel user interfaces for data exploration systems that facilitate creating visualizations and finding the most appropriate ones. Visualizations are just a

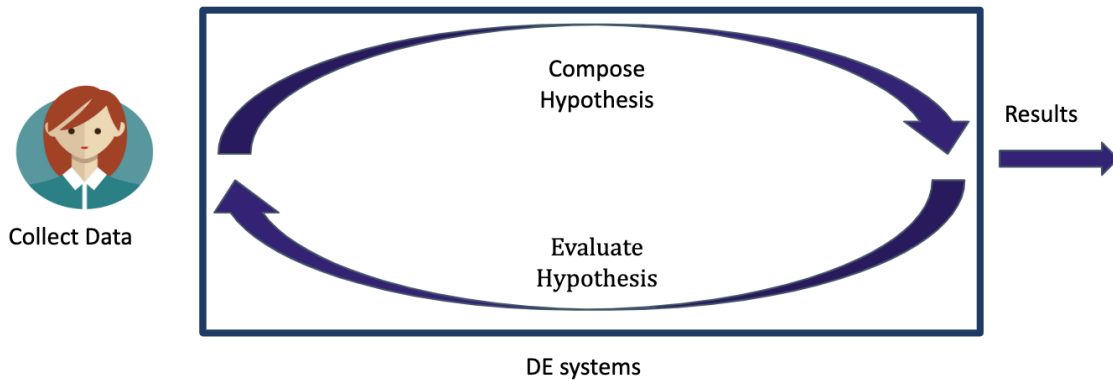


Figure 1.2. How data exploration systems create multiple hypotheses

descriptive statistic of a dataset. As a result, each visualization a user sees at a data exploration system is a different hypothesis test. Some systems [23, 25] make query recommendations to the user during interactive data exploration by extensively searching the query space. Each query is a hypothesis and this search corresponds to testing multiple hypotheses to find the ones that are the most interesting based some criteria. Novel interfaces [24, 43, 60] allow users to explore their data in new ways. New ways that allow users to perform hypothesis testing with just a simple touch on the UI. It is apparent that data exploration systems facilitate data exploration by creating and automatically testing multiple different hypotheses. Figure 1.2 shows how they can test for a lot of hypotheses under the hood to show some results to the user. In addition, when the number of possible discoveries, including false positives, that such a system makes is large, it is hard to follow up on all of them. Even if the system suggests only the most interesting ones, in some research areas it is extremely time consuming to further investigate these probable discoveries. As a result, it is imperative to minimize false discoveries.

Not only do data exploration systems magnify the problem of multiple hypotheses testing, but they also usually ignore its statistical consequences. Unlike traditional DBMSs, data exploration, which is statistics-oriented, operates in a form of the open world assumption, where the data is a mere sample from an underlying unknown distribution, the ground truth. However, most data exploration systems do not use any of the existing statistical methods to make their

discoveries; even when they test for thousands of hypotheses simultaneously. This only makes the problem of false discoveries worse. Systems, such as zenvisage [74], or Polaris [77], attract citizen data scientists and novel users due to their ease of use. Such users have limited statistical knowledge. As a result, they can be lead to honest mistakes of false discoveries due to naivete. In all these cases, controlling for spurious discoveries is imperative.

The problem of multiple hypotheses testing has been studied extensively in the statistics community. Many techniques, such as the Bonferonni correction [16] and controlling the False Discovery Rate (FDR) [12], try to solve the problem. These techniques attempt a trade-off between false positives and true positives in the results. Controlling the FDR has been identified as the most appropriate way of controlling false positives when the number of hypothesis tests is large and in interactive data exploration systems [88]. However, most techniques that control the FDR can still be viewed as conservative for big data applications. Statistical power is used to measure the effectiveness of a testing procedure and is defined as the percentage of discoveries a test finds. Because controlling for false positives can lead to decreased statistical power, current data exploration systems avoid addressing the problem.

1.1.2 Multiple Comparisons Problem in Machine Learning

The multiple comparisons problem has a different flavor in Machine Learning (ML). The goal of ML is to produce models that capture the underlying unknown distribution of the data. These models must generalize well to new data. Each model is a hypothesis that describes the underlying data distribution. Testing a lot of different models can lead to finding one that performs well to the known part of the dataset, but fails to generalize to unseen data. This is a result of the multiple comparisons problem. In ML it is called overfitting and it is a serious hurdle [38].

Overfitting can occur in many different ML processes, when data is used repeatedly, and many of these cases are subtle [53]. To construct a model part of the data is used repeatedly. In

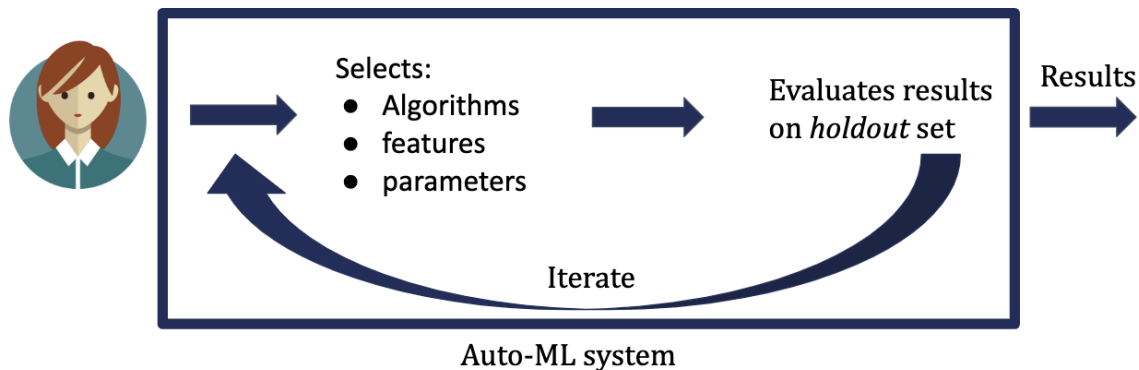


Figure 1.3. How autoML systems create multiple hypotheses

other words, we are testing a lot of hypotheses on the same data. For example, during feature selection the holdout validation dataset is used repeatedly many times to evaluate different models until a good enough is found. The same happens during hyperparameter tuning and generation of base learners for boosting. All these can lead to overfitting the validation dataset [27].

AutoML systems, such as Google Cloud AutoML [2] or Amazon SageMaker Autopilot [1], only make this problem worse. These systems target users who want results with “minimal effort and minimal machine learning expertise” [2]. To achieve that they automate the model selection process, as it can be seen on Figure 1.3. In addition, most non-ML experts usually do not have access to huge datasets, that can help avoid overfitting. For example, at the UCI repository for machine learning datasets, over the past ten years one in three datasets has less than 1000 instances [6]. Most such datasets come from the health sciences. Lastly, lack of machine learning knowledge can lead users to misconceive the results and underestimate the problem of overfitting. For example, when a user sees an accuracy of 95% after putting his data on an autoML system, they might not realize that the system has tested multiple different algorithms, sets of features and hyperparameters and, thus, this high accuracy might be due to overfitting.

Big data might not always help reduce this problem [38]. The abundance of data can

come in the form of more features but not necessarily more sample points for the model to learn from. More features means more models for the feature selection algorithm to test, and with few instances it can easily lead to overfitting [56]. Another very common scenario where big data doesn't necessarily help reduce overfitting is personalized recommender systems [54]. Even though a dataset might have millions of data points of past purchases, only a very small portion (even as little as a few hundred) might be used when learning a recommender model for a particular user. In all these ML cases avoiding overfitting is imperative.

1.2 **VigilaDE: Avoiding False Discoveries with Hierarchical Data**

A lot of real world datasets have a lot of structure. For example, a lot of medical data have well structured ontologies created by experts [26]. These ontologies have hierarchies that, for example, group together similar diseases or pharmaceutical products. This structure of hierarchical data can be used to achieve increased statistical power when controlling for false discoveries.

In this thesis we present VigilaDE (pronounced “vigilante”), a system for Vigilant Data Exploration in the era of big data and citizen data scientists. Data often have some hierarchy that describes the input variables and places them in groups. VigilaDE is the first DE system to exploit hierarchical data in order to guide the data exploration and *avoid testing all possible hypotheses*. VigilaDE's approach is orthogonal to existing controlling techniques. We show that controlling false discoveries is feasible in data exploration systems without losing too much statistical power. VigilaDE achieves up to 2.7x improvement in statistical power against existing controlling techniques.

Contributions. This thesis makes the following contributions:

- Section 3.3 formally defines data exploration of hierarchical data, and how statistical power and false positives are measured in this setting.
- In Section 3.4.1, we present a novel data exploration algorithm for VigilaDE that tries to detect Simpson’s paradox in hierarchical data.
- In Section 3.4.2, we present a novel data exploration algorithm for VigilaDE that guides the exploration by approximating the quality of the input hierarchy.
- In Section 3.5, we present how our algorithms work orthogonal with existing controlling techniques.
- In Section 3.6, we present under which conditions we can create and use a data-driven hierarchy in VigilaDE. We experimentally confirm our findings in Section 3.8.4.
- In Section 3.7, we show through mathematical analysis why and under what conditions VigilaDE leads to increased statistical power.
- In Section 3.8, we experimentally show we can achieve up to 2.7x increase in statistical power with a real-world public health dataset and in simulations.

1.3 AAT: Avoiding Validation Overfitting during Feature Selection

Although there exist many ways in statistics to avoid false discoveries, they cannot be applied in scenarios where hypotheses are chosen adaptively. This scenario was recently identified as being very prominent in ML [27]. Hyperparameter tuning [30], machine learning

competitions [14], generation of base models in boosting or bagging techniques, analyst-in-the-loop systems or fully automated autoML systems and feature selection [67] are all scenarios where the model is chosen in an adaptive way.

In their recent work, the authors of [27] provided some bounds for overfitting and generalization error in this setting. However, their work was theoretical and for a generic setting. They presented ThresholdOut, an algorithm that can answer adaptively chosen queries about a hypothesis and avoid overfitting. However, it requires setting certain parameters that aren't intuitive to the users. In addition, ThresholdOut doesn't take advantage of the *structure of the problem* in specific scenarios, like feature selection. As we show in our work this can lead to significant improvement in performance.

In this thesis we present a novel and superior way to avoid overfitting during feature selection. First, we start by presenting an analysis of the state of the art algorithm that avoids overfitting the holdout dataset in an adaptive setting, called ThresholdOut. Next, we offer an automated way to set its parameters that make it practical and show that it outperforms most manually picked parameter values. Based on what we learn, we present Auto Adjust Threshold (AAT), a novel feature selection algorithm that is inspired by ThresholdOut. AAT takes advantage of the structure of the problem of feature selection, where hypotheses arrive adaptively but in a very specific way. It utilizes the bias-variance trade-off to automatically adjust a threshold throughout the feature selection process, thus, achieving a lower test error.

Contributions. This thesis makes the following contributions:

- In Section 4.3, we present an experimental analysis of ThresholdOut, the theoretical state of the art algorithm.
- In Section 4.3, we give intuitive guidelines for setting the parameters of ThresholdOut.
- In Section 4.4, we present AUTO-SET, a novel and automated way to set ThresholdOut's parameters and we experimentally show that ThresholdOut with AUTO-SET outperforms most manually picked parameters for ThresholdOut.

- In Section 4.5, we present Auto Adjust Threshold (AAT), a novel feature selection algorithm that avoids overfitting the holdout dataset in adaptive scenarios.
- In Section 4.6, we show experimentally that AAT outperforms both ThresholdOut, and ThresholdOut with AUTO-SET.

Chapter 2

Background

In this chapter we give some background knowledge on the topic of statistical hypothesis testing and the existing approaches for controlling for the problem of false discoveries.

2.1 Statistical Hypothesis Testing

A statistical hypothesis test is a procedure that takes as input samples from a population and makes a statement about the parameters that describe the whole population, such as the population mean or the correlation between two variables [18]. That statement comes in the form of rejecting (or failing to reject) the *null hypothesis* H_0 , a hypothesis associated with a contradiction of what one would like to prove. The possible outcomes of a hypothesis test can be seen on Table 2.1.

For example, if we would like to prove that there is a correlation between two variables, then, we would set the null hypothesis H_0 as the fact that there is no correlation between the two variables, as measured by the Pearson correlation ($r = 0$). The alternative hypothesis H_1 is that $r \neq 0$.

More specifically, to test whether the sample data comes from a population in which the null hypothesis H_0 is true, or from a population in which the alternate hypothesis H_1 is true, we

Table 2.1. Possible outcomes in hypothesis testing.

	Alternate Hypothesis is True	Null Hypothesis is True
Test is significant	True Positive	False Positive
Test is non-significant	False Negative	True Negative

use a statistical test, which is a real-valued function $T(x_1, \dots, x_k)$ of the sample data x_1, \dots, x_k . The test statistic is a random variable and we want to calculate the probability of coming from a distribution f_{H_0} , where the null hypothesis is true, or from a distribution f_{H_1} , where the alternate hypothesis is true. f_{H_0} and f_{H_1} are the ground truth that we do not know about our data. We approximate them through the statistical test T and the sample data points we have. In order to make a decision about our hypothesis we need to calculate T 's p-value p_T . p_T tells us how likely it is to observe a result at least as extreme as the test statistic if the null hypothesis is true. The p-value is then used to determine if we can confidently reject or not the null hypothesis. Continuing our example, for each pair $\langle \text{factor}, \text{outcome} \rangle$ of variables we have some example data points. We then calculate the correlation, its t -test as our test statistic and the t -test's p-value p_T for each pair.

We reject the null hypothesis and say that a discovery is statistically significant if p_T is smaller than the significance value α . α is determined before the test and signifies the maximum probability of having a false positive. In other words, a test T is a mapping of the values of the test statistic into $\{1, 0\}$, where 0 implies that we fail to reject the null hypothesis and 1 implies that the null hypothesis H_0 is rejected.

$$Test(T) = \begin{cases} 1 & \text{if } p_T \leq \alpha \\ 0 & \text{if } p_T > \alpha \end{cases}$$

In Figure 2.1, we visualize all that we previously described about how hypothesis testing

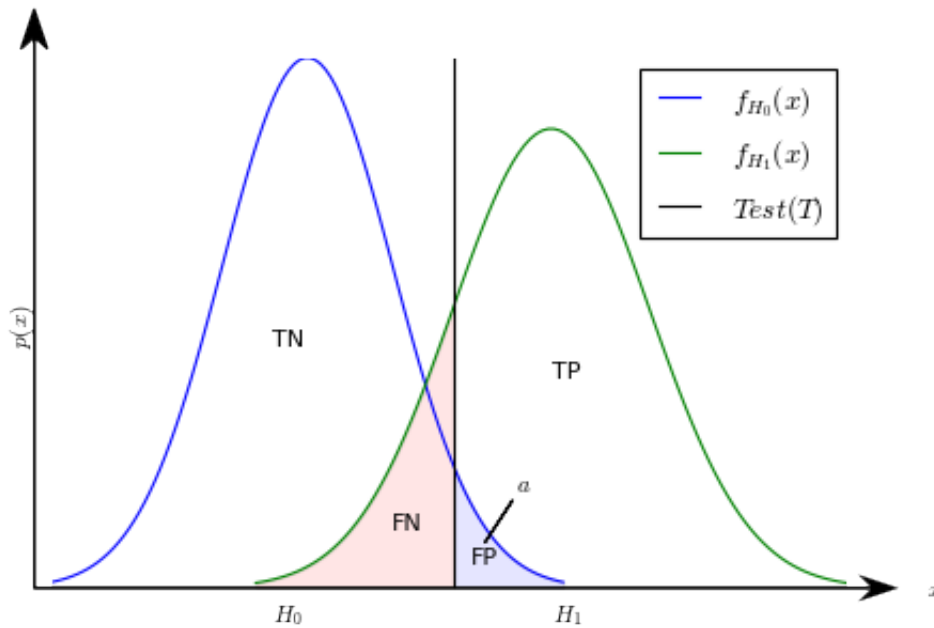


Figure 2.1. Visualizing events and probabilities in hypothesis testing.

works. f_{H_0} is the probability distribution if the null hypothesis were true and f_{H_1} is the probability distribution if the alternative hypothesis were true. Both f_{H_0} and f_{H_1} constitute the ground truth that we approximate through the t -test. More specifically, t -test approximates f_{H_0} and based on p_T (the probability of having a false positive) and the input α we can make a decision if we can reject the null hypothesis, i.e. our sample data do not come from f_{H_0} , or fail to reject the null hypothesis, i.e. we cannot decide if our sample data comes from f_{H_0} or not in a statistically significant way. If we knew the distributions f_{H_0} and f_{H_1} , we could also calculate the probabilities that our decision was correct (true positive/true negative) or not (false positive/false negative).

There are two types of errors that can occur when testing a hypothesis, the *type I and type II errors* (Table 2.1). A type I error (also known as a *false positive*) is the incorrect rejection of a true null hypothesis. A type II error (also known as a *false negative*) is the failure to reject a false null hypothesis. A testing method can minimize the number of false positives by being very conservative in accepting hypotheses, and, thus, having a high type II error.

2.1.1 Multiple hypothesis testing

Multiple hypotheses testing refers to the problem of testing more than one hypothesis at a time. As we test more hypotheses, the chance of spurious discoveries increases. If we use the same α value to reject a null hypotheses for each test, then the probability of making at least one false positive discovery increases.

For example, if we tested 100 different hypotheses simultaneously, for a significance value of $\alpha = 0.05$, under the common assumption that the tests are independent, the probability of having at least one false positive is:

$$\begin{aligned} p(\text{at least one FP}) &= 1 - p(\text{no FP}) \\ &= 1 - (1 - 0.05)^{100} \simeq 0.99 \end{aligned}$$

Even though we do not know the ground truth, we can be almost certain, with a probability of 0.99, that at least one out of a hundred is a false positive. It is obvious that we need to adjust the value of α of each hypothesis to get equivalent guarantees when we test many hypotheses simultaneously.

In order to provide similar guarantees in the case of multiple hypotheses testing, the previous hypotheses testing methods need to be adjusted. The first step in controlling false discoveries in this case is to group hypotheses in *families*. A family is a set of hypotheses for which significance statements will be treated together and the errors will be controlled jointly for the whole family. There are many issues involved with selecting the families, like the fact that the results of an experiment can be manipulated by the selection of the families or the fact that different purposes may require different families of the same data [72].

When testing multiple hypotheses, there are four different probabilistic guarantees that we might want to control. These are:

- *Error Rate per Hypothesis (ERH)*. The ERH is defined as the probability of type I error, or

the expected value of false positives among all the results.

- *Error Rate per Family (ERF)*. The ERF is defined as the expected number of false positives in the family.
- *Familywise Error Rate (FWER)*. The FWER is defined as the probability of having at least one false positive in the family.
- *False Discovery Rate (FDR)*. The FDR is the expected proportion of false positives among the rejected hypotheses.

The last two metrics are the ones that are mostly used in practise.

2.1.2 Statistical power

Statistical power is used to measure the performance of a statistical test when we know the true probability distributions. It is the probability of the test correctly rejecting a false null hypothesis given the alternate hypothesis is true.

$$\begin{aligned} \text{Power} &= p(\text{reject } H_0 | H_1 \text{ is true}) \\ &= \frac{p(TP)}{p(TP) + p(FN)} = p(TP) \end{aligned}$$

Empirically, we can measure statistical power as:

$$\text{Power} = \frac{TP}{TP + FN}$$

2.2 Controlling Multiple Hypotheses

2.2.1 Family-wise Error Rate (FWER)

In multiple hypotheses testing, the chance of occurring a type I error, i.e. a false positive discovery, increases as the number of hypotheses that are being tested increases.

The *Bonferroni inequality* states that for a set of events A_1, A_2, \dots, A_n with probabilities p_1, p_2, \dots, p_n the probability of their union is smaller or equal to the sum of their probabilities.

There exist a number of techniques that are based on the Bonferroni inequality and they try to control the number of false positives in a family. These techniques try to control the level of FWER by testing each individual hypothesis at a much smaller significant level, so that the sum of the smaller significance level of the individual inequalities is smaller or equal to the desired FWER.

Given a set of n hypotheses H_1, H_2, \dots, H_n and their p-values p_1, p_2, \dots, p_n , the most common way to control the FWER based on the Bonferroni inequality is the following: To control the FWER at α , reject the null hypothesis for each $p_i \leq \alpha/n$ [16]. This technique can be generalized so that we control any single hypothesis at any significant level as long as the sum of all the significant levels we use is smaller or equal than α .

Holm, Simes and Hochberg provided some extensions to the basic Bonferroni control procedure which improve the power of the technique under certain conditions while controlling the FWER. However, these methods are still only a minor improvement and are not powerful enough for cases with large number of hypotheses.

Holm provided a sequential rejective method for controlling the FWER based on the Bonferroni inequality [41]. Given a set of n hypotheses H_1, H_2, \dots, H_n and their p-values p_1, p_2, \dots, p_n ordered from smaller p-value to the larger one, at each stage i of the procedure reject H_i if and only if all previous hypotheses have been rejected and $p_i \leq \alpha/(n - i + 1)$. This method increases the significant value we use for each individual hypothesis as their p-value increases and accepts the first k hypotheses. It can be shown that this method controls the

FWER at level a .

Hochberg's procedure is a simple modification of Holm's procedure for controlling the FWER at level a [40]. Given a set of n hypotheses H_1, H_2, \dots, H_n and their p-values p_1, p_2, \dots, p_n ordered from smaller p-value to the larger one, find the hypothesis p_i where $p_i \leq a/(n - i + 1)$ for any $i = 1, \dots, n$ and reject all hypotheses H_j with $j \leq i$.

Lastly, Simes proved a theorem that applied to independent hypotheses and he claimed, through simulation results, that it applied in a few other cases [75]. He suggested that this result could be used in multiple hypotheses testing without providing a formal way to do so. He showed that given a set of n hypotheses H_1, H_2, \dots, H_n that are all true and their p-values p_1, p_2, \dots, p_n ordered, when the test statistics used are independent then $p_i > ia/n$ for $i = 1, \dots, n$ holds with probability $1 - a$.

There are two main disadvantages of the techniques that control the FWER. First, all the methods described here require knowledge of the number of hypotheses that are being tested before hand. This makes all these methods not suitable for interactive data exploration scenarios where the total number of hypotheses is not known until the end of the data exploration. Second, and most importantly, these techniques are too conservative for all data exploration scenarios. The p-value used to reject the null hypothesis gets too small too fast when the number n of hypotheses tests grows large. For example, if we want to control the FWER at a significance level of 0.05, a typical value, for 10 hypotheses we'd test each hypothesis with a significance level of 0.005, for 100 hypotheses we'd use a significance level of 0.0005. It's obvious that this technique is not practical for the case of data exploration where we test hundreds or thousands of hypotheses at a time. Even though Holm, Hochberg and Simes improved the statistical power of the Bonferonni correction, their improvements were minor.

2.2.2 False Discovery Rate

Benjamini and Hochberg proposed controlling the False Discovery Rate (FDR) [11]. The FDR is the expected number of false discoveries among all the discoveries made by a procedure [8]. For example, controlling the FDR at a significance level $\alpha = 0.05$ and if we had 100 significant hypotheses in the result then according to the FDR on average at most 5 of them would be false positives. They proposed this controlling schema as an alternative to controlling the FWER which is too conservative. They showed that controlling the FDR results in significant gain in power of the testing procedure.

The FDR then can be defined as the proportion of false positives as of the total number of significant discoveries. In other words, the FDR is the proportion of true null hypotheses we wrongly reject.

There exists a plethora of ways to control the FDR in different scenarios and with different data [39, 78, 68, 12, 42]. Benjamini and Hochberg proposed a procedure to control the FDR based on a simple derivative of Sime's procedure [75] that was presented earlier.

Given H_1, H_2, \dots, H_n hypotheses and their p-values p_1, p_2, \dots, p_n that are *ordered*, they defined the following rule for the controlling procedure:

$$\text{if } k \text{ is the largest } i \text{ for which } p_i \leq \frac{i}{n}a,$$

Reject all null hypotheses $H_i, i = 1, 2, \dots, k$

They showed that for any test statistic that is independent, the above procedure controls the FDR at level a .

When the tests are dependent the previous threshold is modified to the following:

$$p_i \leq \frac{i}{nc(n)}a$$

When the tests are independent or positively correlated, $c(n) = 1$ and the threshold is equal to the previous one. However, on all other cases $c(n) = \sum_{i=1}^n \frac{1}{i}$. In [12], they showed that this adjusted threshold controls the FDR when the tests are dependent.

Controlling the FDR is less conservative and it is more appropriate when n is large. As a result it is widely used when there is a large number of hypotheses being tested in many different fields [46, 66, 70]. However, it still has lower statistical power compared to not controlling.

2.2.3 *m*FDR and α -investing

α -investing is a family of testing procedures that controls the marginal False Discovery Rate *mFDR*, a derivative of the *FDR* [31]. *mFDR* is the ratio of the expected number of false positives to the expected number of discoveries. The main advantage of α -investing techniques is that they are *adaptive and sequential*. The key idea behind α -investing is that when the testing procedure makes a discovery, the procedure earns additional wealth (i.e. probability to make a false positive in the future) towards the next tests.

α -investing is a set of sequential techniques that can be used for testing multiple hypotheses. These techniques work when tests arrive in batches or sequentially in a stream.

An α -investing rule I is a function that in a sequence of hypothesis tests, it determines the α -level for the next hypothesis test. The initial wealth of the testing procedure is denoted by $W(0)$, and the wealth at step k of the procedure is $W(k) \geq 0$. Common values for the initial wealth are 0.05 and 0.10. At each step i , the α -investing rule I sets the a_i level that the hypothesis H_i will be tested, and a_i could be between 0 and the maximum available wealth $W(i)$.

The outcome of testing all the previous hypotheses determines the available α -wealth $W(j)$ that will be available for testing the hypothesis H_{j+1} . At each step there are two possible scenarios:

- $p_j \leq a_j$. The null hypothesis H_j is rejected and the investing procedure will earn back some α -wealth that is called pay-out, $\omega < 1$.

- $p_j > a_j$. The null hypothesis is accepted and as a result the α -wealth has to decrease by $a_j/(1 - a_j)$.

These two possible scenarios can be summarized in the following equation:

$$W(j) - W(j-1) = \begin{cases} \omega & \text{if } p_j \leq a_j \\ -a_j/(1 - a_j) & \text{if } p_j > a_j \end{cases}$$

So, at each step of the procedure the wealth is either increased when a null hypothesis is rejected, or it is reduced. When the α -wealth of the rule reaches 0, then no further tests are allowed.

This is only one of the many possible payment systems in the family of α -investing rules. Any investing rule with $W(0) = \alpha$, $\omega = \alpha$ that follows the previously described rule controls the m FDR at level $\alpha \in [0, 1]$ [31]. The main difference between different α -investing rules is how to pick the different values for a_i , so as to account for domain knowledge and tailor the testing procedure to some specific task or scenarios.

One advantage of α -investing over all previous techniques is that it is adaptive and sequential. This makes it suitable for interactive scenarios where the number of hypotheses that are tested is unknown. Another advantage of the sequential nature of α -investing is that it allows for more fine tuned control of m FDR. What this means is that the testing procedure will do well even if stopped earlier than the initial scenario. α -investing controls m FDR uniformly. Suppose that we want to stop after testing only 100 hypotheses, instead of n . The uniform control and the sequential nature of α -investing lets us guarantee the average number of false positives that will appear in the result in either case. Lastly, the possibility of picking different investing rules allows the user to capture different application needs. By picking the right investing rule one can account for domain knowledge and tailor the testing procedure to some specific task or scenarios.

Chapter 3

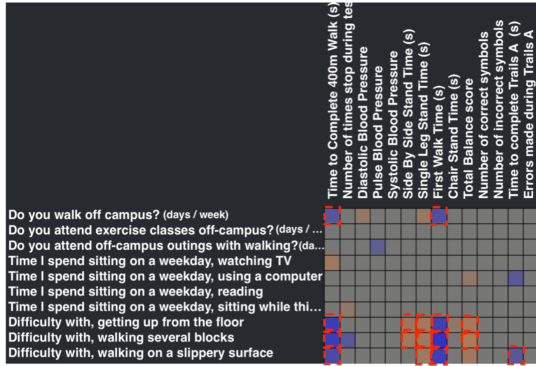
VigilaDE: Avoiding False Discoveries with Hierarchical Data in Data Exploration Systems

In this chapter we present VigilaDE, a system for vigilant data exploration. Hierarchical data appear in many different everyday applications and have a structure that can be used to improve performance. We show how VigilaDE takes advantage of this hierarchical structure to guide the data exploration and reduce false discoveries during data exploration.

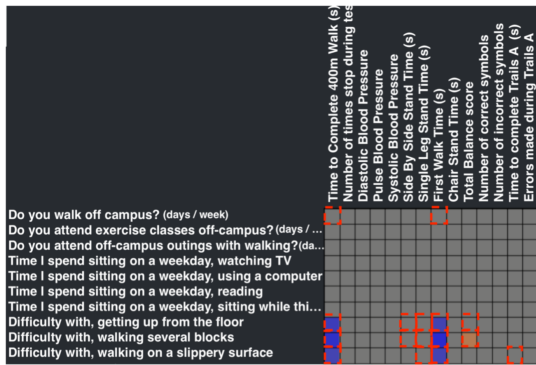
3.1 Motivating Example

In Figure 3.1 we see the results of the data exploration of a real-world public health data set in three different scenarios. Each row represents a factor variable, each column an outcome variable and each square is the visualization of the result of the hypothesis test between the $\langle \text{factor}, \text{outcome} \rangle$ pair that corresponds to that row/column. The blue (or orange respectively) color of a square indicates a negative (or positive) statistically significant correlation. A grey square indicates a non-statistically significant result. This is a common visualization that exists in most data exploration systems, such as in Tableau [79].


In Figure 3.1(a), we see the results of all the hypotheses tests when there is no controlling

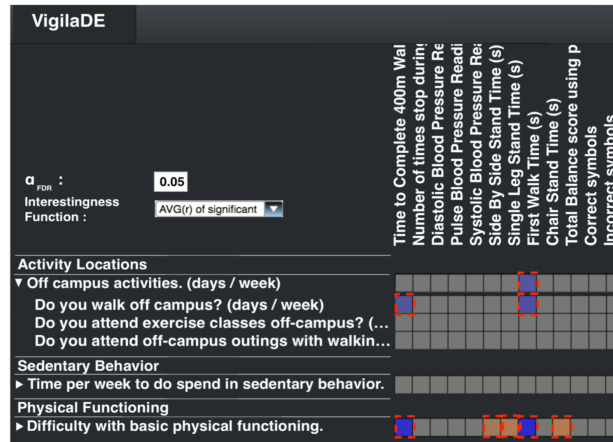


(a) No controlling



(b) Controlling the FDR

 : True Alternate Hypothesis (ground truth)



(c) VigilADE: utilizing hierarchy

Figure 3.1. Visualizing data exploration results of a public health data set. Only part of the data set is visualized for presentation purposes. Blue/orange squares represent a negative/positive statistically significant hypothesis test between the factor and outcome in the corresponding column and row. The opacity of the colors is defined by a user-specified interestingness function.

for the problem of false discoveries. This is what most data exploration systems currently do. In this example, more than a third of the discoveries are spurious; there are 10 false positives among the 26 discoveries. A novice data scientist can easily fall into the trap of thinking that all these discoveries are statistically significant. Without deep understanding of statistics and how the system works, this is an honest mistake.

In Figure 3.1(b), we see the results of exploring the same data set as before, but with applying one of the most widely accepted controlling technique for data exploration [88]. We control the FDR at level $\alpha_{FDR} = 0.05$. To have an *expected proportion* of false discoveries among all discoveries smaller than α_{FDR} , we accept less hypothesis testing results as discoveries. In our example, this translates to an expected number of false discoveries that is smaller than

$0.05 * 7 = 0.35$. In Figure 3.1(b), there are no false positives. However, we get less than half of the discoveries that actually exist in the data (7/16). This translates to a statistical power of only 43% and it exhibits why current controlling techniques can be conservative.

Both scenarios have pitfalls. The first one has too many false discoveries in the results. The second one has low statistical power. We will show that VigilaDE can control false discoveries and have an increased statistical power by taking advantage of the hierarchical structure of the data.

3.2 Overview of VigilaDE

In this Section, we give an overview of VigilaDE. We describe the input hierarchical data, the intuition for VigilaDE's data exploration algorithms, how it fits in with existing controlling techniques and, finally, our experimental results.

Hierarchical Data. Many real-world data sets have some hierarchy that places input variables together in groups. These hierarchies are usually defined before collecting the data and capture semantic knowledge of field experts. For instance, a lot of medical data have well structured ontologies created by experts [26]. CDC uses questionnaires with hierarchical structure to collect information about individuals [22]. In our example, in Figure 3.1(a), if we examine the rows, we can see some semantic structure; most of the rows describe similar factors. This is captured by a hierarchy that was created by scientists and is used in VigilaDE.

VigilaDE's Approach. These groups are not created randomly. Variables that appear in a group together are expected to behave similarly. For example, people who have problems with their sleep (1st variable) also do not feel refreshed after sleeping (2nd variable). We can expect these two variables to be correlated with the same outcomes as a group variable that summarizes them (low sleep quality; 3rd variable). As a result, the discovery that the child variables (1st and

2^{nd} variables) are associated with an outcome does not provide much new information over the discovery that the group variable is associated with the same outcome. Intuitively, we can exploit the input hierarchies to avoid testing some variable pairs altogether. This can lead to finding less spurious discoveries.

VigilaDE's novelty lies in being the first data exploration system that formally exploits this intuition for hierarchical data. Its goal is to avoid testing all possible pairs of input variables. Testing a larger number of such pairs can lead to more spurious discoveries. To achieve that, it uses the a priori knowledge provided by hierarchies to guide the data exploration in a top-down approach. As finding the optimal solution is not possible, VigilaDE has two different data exploration algorithms that try to determine which pairs are less likely to provide new information based on heuristics. One heuristic captures the previously described intuition; variables in the same group are expected to behave similarly. It quantifies this by approximating the quality of the hierarchy and then uses it to guide the exploration of the hierarchical data. Both data exploration algorithms are described in Section 3.4.

There is a plethora of ways to control false discoveries in the statistics community even by utilizing the hierarchical structure of data in some ways. In Section 3.5, we discuss why the chosen controlling technique is *orthogonal* to the data exploration algorithms that avoid performing all hypothesis tests. We demonstrate this experimentally as well, by comparing against an existing technique that utilizes hierarchies.

It is important to note here that these input hierarchies that VigilaDE utilizes are a priori created hierarchies, and *not data-driven*. Input hierarchies are usually preferred over data-driven ones as they already exist in many fields and have semantic structure that is necessary for interpretability. On the other hand, data-driven hierarchies have limitations, such as that variables have to be of the same data type to be aggregated in any possible grouping. In Section 3.6 and 3.8.4, we present and experimentally evaluate the limitations and the conditions under which data-driven hierarchies can be beneficial.

Example. In Figure 3.1(c), we see the results of our public health data set in VigilaDE. It uses the input hierarchy that was created by the scientists before collecting the data to guide the data exploration in a top-down way. We see that all the previous factors are now represented by just 3 group factors (rows 1, 5 and 6). VigilaDE’s algorithm decided to expand and test all the child variables in the first group only. It avoided testing the hypotheses between the child variables of the second and third group with all outcomes. The FDR was controlled at $\alpha_{FDR} = 0.05$ and in this example we have 0 false discoveries. Most importantly, VigilaDE marks as significant all 8 out of 8 true discoveries that exist among the pairs that it chose to test and show in the UI. This results in a significant increase in statistical power.

Experimental Evaluation. Simulations are a very common and statistically sound way to evaluate methods for controlling false positives [8, 12]. First, we use various simulations to show that our techniques achieve a gain in statistical power in most cases, in some cases up to 2.7x. We compare our approach against two existing controlling techniques, one agnostic of any hierarchical structures and another one that utilizes hierarchical data. Next, we validate our approach against a real-world public health data set and show that there exist real-world data sets that exhibits the characteristics we describe in our work. In this particular data set we can achieve up to 1.8x improvement in power.

3.3 Problem Definition

Input variables. Given a set of variables $I = \{v_1, \dots, v_n\}$ that describe some underlying population, the input data set consists of values for the variables in I that describe a sample population. We will call *factor variables*, or simply *factors*, the subset of k variables $F = \{f_1, \dots, f_k\}$ that affect the l *outcome variables*, or simply referred to as *outcomes*, $O = \{o_1, \dots, o_l\}$. The union of the factors and the outcomes consists the input set of variables $I = \{v_1, \dots, v_n\} =$

$$\{f_1, \dots, f_k\} \cup \{o_1, \dots, o_l\}.$$

In our motivating example, the sample is elderly people who took part in the study. Factors describe individual characteristics, such as level of weekly activity, and are represented as rows in Figure 3.1. Outcomes describe health-related characteristics and are represented as columns.

Hierarchy. As part of the analysis process, experts already capture the semantic structure that appears in data in many fields. In our running example, public health experts created questionnaires with hierarchical structure to collect information about individuals, as it can be seen in Figure 3.1. In general both factor and outcome variables can have such a hierarchical structure. This hierarchical structure in our problem is formally captured by a hierarchy D that is part of the input. Formally,

Definition 3.3.1 *Hierarchy.* A hierarchy is a graph $D = (I, E)$, and more specifically a set of rooted trees. Input variables I (factors and outcomes) are vertices in D . E edges represent how the input variables connect to each other.

Based on the definition of a hierarchy D as a graph, we give a few more definitions that we use to reason about.

$D_{leaf}(v)$ is the set of variables of vertex degree 1 in the graph with root v , except for the root. $parent(v)$ is a neighbor of v along an edge that is on the path to the root. $child(v)$ is a vertex of which v is the parent. $descendants(v)$ is the set of vertices which are either the children of v or are (recursively) the descendants of any of the children of v .

Internal variables are an aggregate of leaf variables, that is why we also call them *aggregate* variables in our setting. This aggregate is not the same for every internal variable. First, it is best to be decided by domain experts as in different scenarios different aggregates could have more semantic meaning (i.e., *average* time spend in sedentary activities and *maximum* time exposed to any toxic chemicals). In addition, for all variables to have the same aggregate function, they would have to be of the same type which is a big limitation. For example, in

Table 3.1. Notation used in this paper.

Symbol	Meaning
H_0/H_1	Null/Alternative hypothesis
f_{H_0}/f_{H_1}	True probability distributions of H_0/H_1
p_T	p-value of statistical test T
α	Input significance level
α_{FDR}	Significance level when controlling FDR
F/O	Factor/Outcome input variables
$D = (I, E)$	Input hierarchy D as graph
$D_{leaf}(v)$	Leaves of hierarchical graph with root v
$R = R_f \cup R_o$	Output set of factors/outcomes

our public health data set there are binary variables for yes/no answers and different kind of categorical variables (i.e., how often in a week with 0-7 values, or how much do you agree with a statement with 1-5 values).

Output. The output $R = R_f \cup R_o$ consists of two sets $R_f = \{r_1, \dots, r_k | r_i \in F\}$, $R_o = \{r_1, \dots, r_l | r_i \in O\}$, which are *subsets* of the input variable sets $R_f \subseteq F, R_o \subseteq O$. These sets need to satisfy the following two constrains:

- Any variable in the output cannot be a descendent of another variable in the output:
 $\forall r_i, r_j \in R : r_i \notin \text{descendents}(r_j)$.
- The set of all descendents of all variables in the outcome is equal to the set of leaf-level input variables: $\bigcup_{r_i \in R} \text{descendents}(r_i) = \bigcup_{r_i \in \text{roots}(G)} D_{leaf}(r_i)$.

The output sets are what is visualized in a data exploration system, as in Figure 3.1(c). R_f is visualized as rows and R_o as columns, respectively.

It is important to clarify what we mean by controlling the number of false positives in hierarchical data. We are concerned by the number of false positives that appear in the result R_f, R_o , whether it is between input leaf or aggregate variables. Even though aggregate variables summarize a number of leaf variables, when we control false positives at a level α , aggregate and leaf variables are treated the same. Thus, a true discovery might associate an aggregate variable

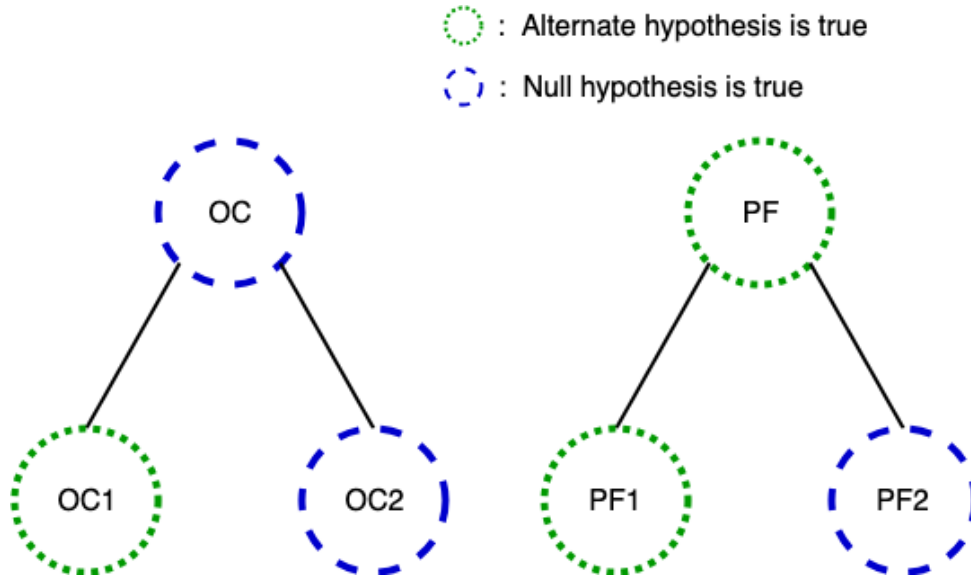


Figure 3.2. Small part of input hierarchy presented in Figure 3.1. We see the results of the true probability distribution between each factor shown and one outcome. **OC:** Off campus activities. **OC1:** Do you walk off campus? **OC2:** Do you attend exercise classes off campus? **PF:** Difficulty with basic physical functioning. **PF1:** Difficulty with getting up from the floor **PF2:** Difficulty with walking.

with an outcome even when not all child variables are associated with the outcome. In Figure 3.2 for the output set $R_f = \{OC, PF1, PF2\}$, if our testing procedure declared significant either OC or PF2, either one would count as one false discovery in the result set. However, in this case, we might lose some facts about the child variables of OC. This is captured by the extended statistical power we define next.

Statistical Power for Hierarchical Data. We need to extend the definition of statistical power for when we utilize hierarchical data. For example, in Figure 3.1(c) statistical power is 100%, even though we lose some information by selecting to show only part of the input variables and not performing all hypothesis tests. That is because the definition of power does not distinguish between leaf and aggregate variables.

Definition 3.3.2 *Extended Statistical Power.* *Extended statistical power is the fraction of true positives divided by true positives and false negatives between the leaf child variables of the variables that appear in the result sets R_f, R_o .*

$$\frac{\sum_{u_i \in D_{leaf}(v_i), u_j \in D_{leaf}(v_j)} P(TP(u_i, u_j))}{\sum_{u_i \in D_{leaf}(v_i), u_j \in D_{leaf}(v_j)} (P(TP(u_i, u_j)) + P(FN(u_i, u_j)))}$$

where $p(TP(u_i, u_j))$ is the probability of having a true positive for the hypothesis test between $u_i \in R_f$ and $u_j \in R_o$.

When we know the underlying true probability distributions, the probability of having a TP or FN is simply either 0 or 1. Whenever we mention statistical power for hierarchical data in the rest of our work, we will mean this extended definition of statistical power.

Using this definition, statistical power in Figure 3.1(c) is $15/16 = 93.75\%$, not 100%. In general, this extended definition of statistical power is necessary to compare our technique against techniques that do not utilize the hierarchical structure of the input data, as in Figures 3.1(a), 3.1(b) and 3.1(c).

Extended statistical power tries to capture the information loss we incur by not performing all hypothesis tests. There are two kinds of information loss we can have when we pick an aggregate variable v_{ag} over some child variables $v_i, i = 1, \dots, n$. Either v_{ag} is not a significant discovery and a child variable v_i is, or the other way around. The first case is captured by the extended definition of statistical power. For example, in Figure 3.2, for an output set $R_f = \{OC, PF\}$ let's assume that our testing procedure correctly identifies as significant only PF. Then, we are losing the information that OC1 is a discovery. Statistical power is 100%, whereas extended statistical power is 50%. In the latter case, even though some child variable v_i might not be a discovery, the effect of the aggregate variable v_{ag} is strong enough so that it is an actual true discovery. This can be seen in PF aggregate variable in the previous example. What we are losing in this case is that some sub-population of the group might not be associated with the outcome (PF2) and some other sub-population might be associated more strongly (PF1). The net outcome is that on the group level, v_{ag} is still associated with the outcome. If the user is more interested in a specific group level discovery, they can pick to explore it further.

Heuristics. Given two outputs R_1 and R_2 , where the false discoveries are controlled at a given level α , we want to pick the output set which has the highest extended statistical power.

Finding an optimal solution has a caveat. Knowing the statistical power of a solution requires performing every possible hypothesis test, which is what we are trying to avoid. It also requires knowing the true probability distributions of these hypotheses. To solve this problem we present different *heuristics* in our data exploration algorithms. These heuristics try to approximate the quality of the hierarchies.

3.4 Data Exploration Algorithms

In this section we present VigilaDE’s data exploration algorithms. Specifically, we present two algorithms that pick R_f, R_o , the variables that appear on Figure 3.1(c).

3.4.1 Data Overview Algorithm

Data Overview Algorithm picks to expand groups whose aggregate variables are closest to being a discovery with the hope that many tests with child variables will be discoveries.

Intuition. Users want to get the most information when they first visualize their data sets in a data exploration system. That is why we want to pick R_f and R_o that maximize statistical power. The first heuristic tries to detect discoveries that disappear when dissimilar variables, with respect to certain outcomes, are grouped together. In particular, several variables in a group might be correlated with an outcome, but when these variables are combined, the group variable might no longer be correlated with the outcome. This is known as Simpson’s paradox [15] and appears often when the groups do not contain variables that behave similarly (i.e., we have a hierarchy of low quality). Our heuristic tries to approximate this by picking the group variable

with the *smallest non-significant p-value* as most likely to be a result of Simpson’s paradox. For example, between two group variables with p-values 0.06 and 0.5, it is more likely that some of the variables in the first group are discoveries and, thus, bring the group ‘closer’ to being a discovery. So, we expect to find more discoveries and gain more power by expanding the first group variable. In our case, each such group variable is associated with many other variables, so we use as a heuristic the *smallest average p-value of non significant tests*.

Example. In Figure 3.2 we see one case where some child variable is associated with the outcome but that is lost on the group level. We can see that OCI is correlated with O but OC is not. This can happen because the effect of OCI is not strong enough to make OC a discovery when $OC2$ is not a discovery. In other cases, this could happen when not all child variables are correlated with the outcome in a significant way.

Algorithm. The algorithm starts with all factors F , outcomes O , the hierarchies D_f, D_o and k_f, k_o the maximum number of factors and outcomes that will appear on the initial UI of VigilaDE. k_f, k_o are the stopping criteria for our algorithms and are picked by the user. They should be set between the number of all leaf-level variables and the number of the root-level variables in the input.

This algorithm requires to have a sequential technique for controlling false positives because at each step the heuristic needs to pick the non-significant tests.

The output of this algorithm is two sets of variables R_f, R_o and the result of the hypothesis test for each pair $\langle \text{factor}, \text{outcome} \rangle$, $factor \in R_f, outcome \in R_o$.

The algorithm starts with the top level group variables from D_f, D_o . It places these in a candidate set C . Then it performs the hypothesis tests for all pairs of $\langle \text{factor}, \text{outcome} \rangle$ for factors and outcomes in C and adds them to R_f, R_o . Next, it finds the group factor or outcome with the smallest average p-value of non statistically significant tests and it expands it, if by expanding it we don’t get more than k_f or k_o variables in the results. It expands that variable and adds its child variables in the candidate set C and the result set R_f or R_o . The algorithm ends

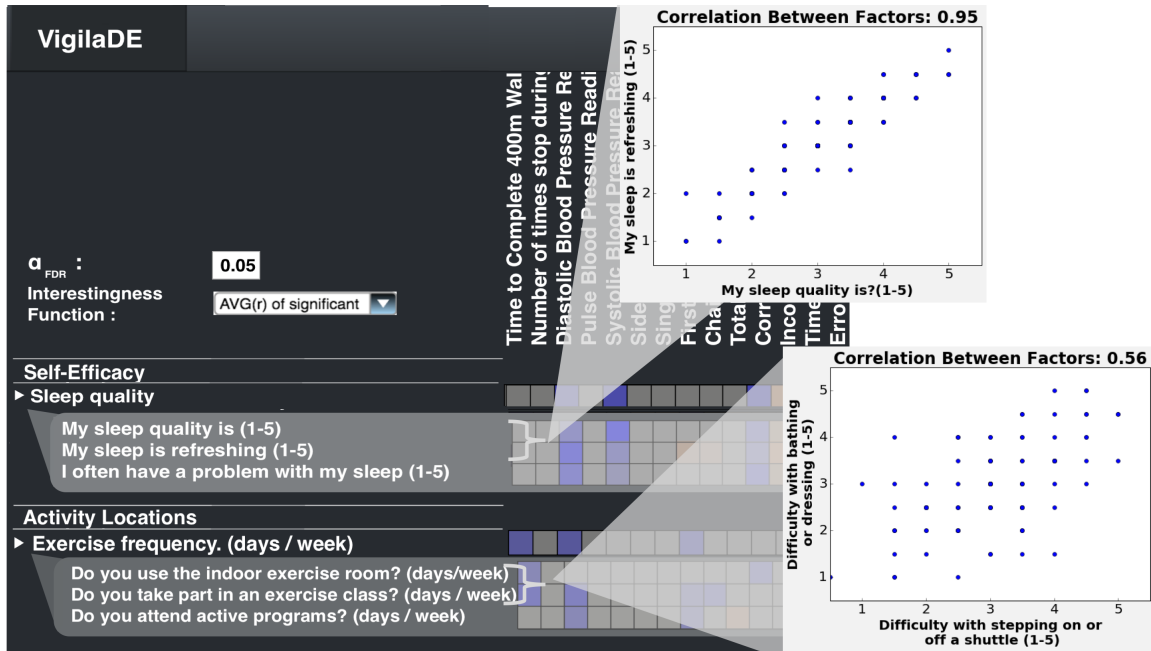


Figure 3.3. Visualizing the correlation between child variables of two different group factors between which the Group Quality algorithm has to pick.

when it can no longer expand any variables, either because it has reached the limits of k_f, k_o or because all variables are leaf-level variables. Then it returns R_f, R_o .

3.4.2 Group Quality Algorithm

Group Quality Algorithm tries to approximate group quality. It does that without performing the actual hypothesis tests between child variables but by just seeing how correlated with each other these child variables are. We expect this heuristic to perform best when the quality of the hierarchy is not ideal, which is most real world scenarios.

Intuition. The quality of the hierarchy affects the gain we can get from testing all the child variables over the parent variable. The quality of the hierarchy describes how similar is the relationship between the group variable and the outcomes and the relationship between the child variables and the outcomes. A group with low quality can offer a substantial increase in power. As a result, when the quality of the hierarchy is low, we expect to gain a lot of power

by using this quality heuristic. When child variables in a group are not similar, we get different discoveries in the child variables that are lost on the parent level. As calculating the actual quality of the hierarchy requires performing all hypothesis tests, we use a heuristic to approximate it.

Example. Intuitively, if two factors in the same group behave similarly, then they might behave similarly when it comes to outcomes as well. People who have problems with their sleep (first child factor) usually do not feel refreshed after sleeping (second child factor). So, we would expect that they would be associated with the same outcomes as people with low sleep quality (group factor). In this case, our hierarchy has good quality and we expect not to gain statistical power by testing the child variables. We visualize this in Figure 3.3. We see 2 group factors, sleep quality and exercise frequency. We see the correlation between one pair of child factors in each group. We chose the correlation as a metric of similarity in our data set. The first group has an average correlation 0.95 between all child variables, whereas the second one 0.56. As a result, we expect to gain more information by expanding the second group variable. We can see on the figure that in the second group there are more factors that are correlated with only a few outcomes in contrast to the first group where all the factors are correlated with mostly the same outcomes. In this example our heuristic achieves its goal. There are many cases where it would fail. For example, if feeling unrefreshed after sleeping (child variable) was caused by a condition such as chronic fatigue syndrome (outcome) that does not cause problems in general with sleep (child variable), our heuristic would fail to detect this. The child variables are correlated with each other, so our heuristic would think that the group quality is high. But, because some outcomes might be associated with only one specific child variable, we would lose power by not selecting to explore this group.

Algorithm. Formally, *group quality* is the metric we use to capture the similarity between variables in a group. We define it as the average of the pairwise Pearson correlation:

$$Group\ quality(G) = \frac{\sum_{v_i, v_j \in G} sim_{v_i, v_j}}{n_{ch}(n_{ch} - 1)/2} = \frac{\sum_{v_i, v_j \in G} |\rho_{v_i, v_j}|}{n_{ch}(n_{ch} - 1)/2}$$

where G is the group of which we are measuring the quality, n_{ch} the number of child variables in the group, sim_{v_i, v_j} a similarity metric between two sample variables, and ρ is the Pearson correlation coefficient, the metric we picked for our problem. We chose the Pearson correlation to measure the similarity as it is invariant to scaling and adding of constant factors. This choice does not relate to using correlation as part of the hypothesis tests we perform in this data. A group quality of 1, means that all child variables are perfectly correlated with each other and is the ideal quality. The lower the quality gets the more potential to gain statistical power.

The group quality can be used as criteria for picking which group variable to expand in the data exploration algorithm we previously described. In each step of the algorithm, we expand the candidate that has the lowest quality.

3.5 Controlling in VigilaDE

Our proposed algorithms are orthogonal to existing controlling techniques and offer the versatility of picking the most appropriate controlling technique in each scenario. However there are certain factors that might affect the performance or that pose restrictions on which method to pick to control false discoveries. In this section we examine these factors.

3.5.1 Orthogonality

Group structures. Group structures have been used to control false discoveries in different ways. Input groups can be used to weigh the significance value α_{FDR} among different groups based on some estimate of true discoveries [42]. This results in increased statistical power when true discoveries appear together in groups. This method is orthogonal to our algorithms. We used it as one of our baselines in the experiments and noticed that our algorithms can significantly improve power when the quality of the groups is low, i.e. when true discoveries do not appear

together in groups.

Two-stage and stratified approaches control the FDR separately within each group as some groups might have a much larger number of discoveries and this can lead to increased power [78, 87]. Such techniques can be easily applied in VigilaDE and can possibly lead to an increase in power.

Heller et al identified that performing hypothesis tests on clusters of data can result in fewer tests with larger populations [39]. They suggested creating groups based on an external data set with field specific features, such as time and space properties of fMRI data, and then performing hypothesis testing with some existing technique that uses these groups. In the next section we discuss that data-driven hierarchies have certain limitations in our setting. In our approach we do not even need an external data set with features to perform clustering because we are utilizing the group quality metric we define in this paper.

Data Dependence. Controlling correctly the FDR depends on the chosen method and the assumptions of the method about the data dependence between all hypothesis tests. There are many different approaches to controlling the FDR when there is some kind of dependence in the data [68]. The original FDR procedure proposed by Benjamini and Hochberg was shown to still control the FDR when there is a positive regression dependency (PRD) among the test statistics [12]. However, when there is no positive dependence or the dependence is unknown, Benjamini and Yekutieli proposed a more conservative way to control the FDR [12].

Under PRD, an increased expected p value for a group true null hypothesis must not lead to a decrease in the expected p value of any of the children true null hypotheses. The input hierarchical structure of our data implies such dependence among the group variable and its child variables. As a result, we argue that using the original FDR procedure, and any other controlling method that utilize the PRD to ensure the controlling guarantees (i.e., [42]), controls the FDR correctly in our algorithms. When there is an unknown dependence in the data the technique proposed in [12] controls the FDR correctly. Our algorithm will still yield a significant benefit in

statistical power even with such conservative approaches.

3.5.2 Interactive Data Exploration

After the user is presented the initial results of the data exploration algorithms, she can choose to further explore the data in an interactive way. In order to support interactive data exploration we need a sequential controlling technique. In such cases, we can chose to control the $mFDR$ using α -investing and adjust the investing rules proposed in [88].

α -investing controls the $mFDR$, a variation of the FDR . In VigilaDE, we need to adjust the proposed investing rules. When we expand a group variable v_g , we perform $n_{child}m$ tests, where n_{child} is the number of child variables that v_g has in the hierarchy D and m is the number of factors/outcomes we currently have in R_o/R_f and we have to test v_g against. To adjust the investing rules, we batch all $n_{child}m$ together and divide the significance level α_j of each step of the α -investing rules uniformly over the $n_{child}m$ tests.

3.6 Data-driven Hierarchies

So far we have focused our work in exploring how to use hierarchies that already exist in many fields to improve data exploration. In certain fields, such hierarchies do not exist. In this section, we explore under what conditions such a hierarchy could be created and used from the input data.

To create a data-driven hierarchy, we can use any clustering algorithm and combine variables based on a similarity function, such as their correlation. As a result, this groups will have a high group quality for our algorithms. However, selecting the right clustering algorithm and setting its hyperparameters, like the number of clusters we want, might affect the quality of the final data-driven hierarchy. More importantly, for the data-driven approach to work the same

aggregate function has to be used to create all aggregate variables automatically. For different variables in the input, different aggregate variables might make more sense. For example, the two functions: *AVG* and *MAX* cannot be used at the same time in a data-driven hierarchy, even though there might be some groups where the first one is more appropriate and groups where the later is more appropriate. One such example is the average time spend in sedentary activities and the maximum time exposed to any toxic chemicals.

Another advantage of utilizing existing hierarchies is interpretability, which is very important for DE systems. These existing hierarchies usually come with group variable names that summarize semantically the child variables in each group. These are the names that would be used on a DE system, such as in Figure 3.1(c). On the other hand, for clustering algorithms it is hard to generate such a semantically meaningful name. Furthermore, they can group together seemingly unrelated variables (which can be argued that is of interest to the user in other applications). For example, for our example data set we run k-means clustering algorithm on the input factors (with $k = 27$, as many clusters are there exist in the human created hierarchy). Variables such as “During the past week, I felt fearful.” and “Do you attend exercise classes off-campus? (days/week)” appeared in the same cluster. Such a grouping isn’t easily interpretable.

Data-driven hierarchies are an alternative to using preexisting hierarchies under the following conditions:

- There is no preexisting hierarchy, or its quality is low, or we are unsure of its quality.
- We are not interested in using different aggregate functions for the same type of input variables.
- We are not interested in interpretability of the groups for application purposes.

In Section 3.8.4, we explore experimentally when data-driven hierarchies can be used in a beneficial way with our approach for data exploration.

3.7 Analysis of Gain in Power

In this section, we use principled mathematical analysis to show formally the reasons that affect the potential gain in statistical power. We show that the effect of a factor on the outcome, a higher sample size and the quality of a hierarchy affect the possible gain in statistical power.

We assume that all input variables are discrete. Without loss of generality, we assume that we have boolean variables. Non-boolean discrete variables can be easily converted to multiple boolean variables. All the following conclusions and intuitions that we prove hold for continuous variables as well. However, do to space limitations we have omitted our proofs for continuous variables as they are almost identical. However, with discrete boolean variables the probabilistic part of the analysis is easier to follow and more intuitive.

Analysis for Hierarchical Data. In hierarchical data, given a group factor f_g and its n_g child variables f_i , we want to examine when testing only the pair $\langle f_g, o \rangle$ over testing all n_g pairs $\langle f_i, o \rangle$ provides more statistical power.

$$Power_g \geq Power_{children}$$

Where $Power_g$ is the power of the procedure that tests only the group factor with the outcome and $Power_{children}$ is the power of the procedure that tests all n_g children.

From the definition of statistical power we get:

$$Power_{children} = \frac{\sum_{n_g} p_i(TP)}{\sum_{n_g} [p_i(TP) + p_i(FN)]} = \frac{Power_i}{n_g} \iff$$

$$Power_g \geq \frac{Power_i}{n_g} \tag{3.1}$$

where $p_i(TP)/p_i(FN)$ is the probability of the test on the i -th child variable to be a TP/FN, $Power_i$ is the power of a procedure that tests only child i and $p(FN) = 1 - p(TP)$.

The hypothesis H_1 that we want to test is whether when one has a factor variable f , they are more (or less) likely to have the outcome o . In other words, we want to see if f is correlated with o . The statistic that we are interested in is the probability $p(o|f)$. The null hypothesis H_0 that we try to reject is that having f does not affect the probability of having o , i.e., $p(o|f) = p(o)$.

The appropriate test statistic for this case is the z-test [76]:

$$z = \frac{p(o|f) - p(o)}{e} = \frac{p(o|f) - p(o)}{\sqrt{p(o)(1 - p(o))/n_s}}$$

where e is the standard error of $p(o)$ from n_s sample objects.

It is preferable to test $\langle f_1, o \rangle$ over $\langle f_2, o \rangle$ when the first hypothesis is more likely to result in a true positive. This is measured by the statistical power. The alternate hypothesis is that the correlation is not zero, $\rho \neq 0$, so we need a two tailed z-test. The null hypothesis is rejected when the $|z|$ value of the test is bigger than $z_{critical}$, which is determined by the significance level α . As a result in equation 3.1:

$$Power_g \geq \frac{Power_i}{n_g} \iff |z_g| \geq \frac{\sum_{n_g} |z_i|}{n_g}$$

By replacing the z values from the previous definitions:

$$\begin{aligned} \iff \frac{|p(o|f_g) - p(o)|}{e_g} &\geq \frac{\sum_{n_g} \frac{|p(o|f_i) - p(o)|}{e_i}}{n_g} \\ \iff \frac{|p(o|f_g) - p(o)|}{\sqrt{p(o)(1 - p(o))/n_g}} &\geq \frac{\sum_{n_g} \frac{|p(o|f_i) - p(o)|}{\sqrt{p(o)(1 - p(o))/n_{s_i}}}}{n_g} \end{aligned}$$

$$\begin{aligned}
&\Leftrightarrow \frac{|p(o|f_g) - p(o)|\sqrt{n_g}}{\sqrt{p(o)(1-p(o))}} \geq \frac{\sum_{n_g} |p(o|f_i) - p(o)|\sqrt{n_{s_i}}}{n_g \sqrt{p(o)(1-p(o))}} \\
&\Leftrightarrow |p(o|f_g) - p(o)|\sqrt{n_g} \geq \frac{\sum_{n_g} |p(o|f_i) - p(o)|\sqrt{n_{s_i}}}{n_g} \tag{3.2}
\end{aligned}$$

Takeaway. From equation 3.2, we see that picking the best option is affected by two factors: the square root of the sample size that supports each factor and the (average) effect a factor has on the outcome. The fraction of the effect of the group factor to the average effect of its child factors is the *quality* of the hierarchy. In a perfect hierarchy, all child factors would have the same effect on the outcome as the group factor. In that case, there would be no benefit in testing the child variables. We could gain power because we would perform less hypothesis tests, and we would have a bigger sample size. Expanding a group with high power doesn't provide us with much new information. However, as the quality of a hierarchy becomes lower we can gain power by testing the child variables. As a result, if all other factors were the same, between two groups we would pick to expand the one with the lowest quality to gain the most power. This is the main intuition behind our algorithm in Section 3.4.2.

3.8 Experimental Evaluation

In this section, we show experimentally with real world data and simulations that our techniques can achieve a significant gain in statistical power in various data exploration scenarios while controlling false discoveries.

3.8.1 Experimental Design

Simulations are a very common and statistically sound way to evaluate methods for controlling false positives [8, 12]. For our simulations, we created m factor and n outcome random variables. Each variable is a sample of observations from a normal distribution $\mathcal{N}(\mu, \sigma)$, where μ is either $\mu_1 = 0$ or $\mu_2 = 5/4$ with probabilities p_1 and p_2 respectively. Each variable has a sample size s , that unless stated it is the same for all the variables in each experiment. The hypothesis that we are testing is whether a pair of $\langle factor, outcome \rangle$ variables comes from a different probability distribution. The null hypothesis we are trying to reject is that the two variables come from the same normal distribution. Probabilities p_1, p_2 and n, m determine the number of true null hypotheses that exist in the data ($p_1^2 nm + p_2^2 nm$). We used a two-sided t-test to test our hypotheses at a significance level $\alpha = 0.05$, which is a common value. For a number n_c of child variables and quality q , we place qn_c variables that come from the same distribution in each group in the hierarchy. All simulations are repeated 100 times and we report averages and standard deviation.

Baselines. For our experiments we used two different baselines, depending on the controlling technique that we used. The first one is independent of the input hierarchy and its quality, whereas the second one takes advantage of the input hierarchy and, as a result, its performance depends on its quality. First, we controlled for the FDR using the Benjamini-Hochberg procedure. In this case, the *baseline* is to perform all possible hypothesis tests and control the FDR. We compare this baseline against the performance of our algorithms when we control the FDR with the Benjamini-Hochberg procedure. Second, we controlled the FDR with the Group Benjamini-Hochberg (GBH) procedure proposed in [42] which takes advantage of the hierarchy. In this case, the *baseline* is to perform all possible hypothesis tests and control the FDR with GBH procedure. We compare this baseline against the performance of our algorithms when we control the FDR with the GBH procedure as well.

3.8.2 Statistical Power Gain Against Baselines

In this experiment we show that for different controlling techniques, our approach can achieve a significant increase in statistical power that can be up to 2.7x. We created different data sets and we tested our approach against both baselines. For $m = 1000$, $n = 1000$ we varied p_1 and p_2 so that we get a different number of discoveries. We also varied the quality of the hierarchy to examine how it affects statistical power. We picked k to be 10% of the number of factors and the number of outcomes, $k = k_f + k_o = 100 + 100 = 200$. We see all our results in Figure 3.4.

In Figure 3.4(a), we see that in all cases we achieve some improvement in statistical power. This improvement can be up to 2.7x (from 0.23 on average to 0.86). In certain cases, the power of the baseline is low, and the data favor our approach to achieve a very high statistical power. We see that when the number of true alternative hypotheses is higher, the first baseline detects more of these hypotheses and has a higher power. So, in those cases the potential increase in statistical power is lower. We also see that, as expected, when the quality of the hierarchy gets lower then so does the gain in statistical power. Last, we see that the Data Overview algorithm performs best when the quality of the hierarchy is high as it assumes that the hierarchies are of high quality and then tries to find which groups to expand. When the quality gets lower Data Quality algorithm starts to outperform it as it tries to identify the groups that have low quality, and it pays off.

In Figure 3.4(b), we see the results of the second baseline and our algorithms with controlling the GBH. We can still that our approaches still achieve a big gain in statistical power and our previous observations about when each algorithms performs best still hold. What is interesting to note is that the performance of GBH depends on the data quality, as it needs to predict how many true alternative hypotheses exist in each group. On the other hand, controlling the FDR with the BH procedure is independent of data quality. As a result, even though it might seem that it is always beneficial to use the GBH procedure to control for false discoveries (with or

without our algorithms), when we are not aware if there exists any locality among the alternative hypotheses (i.e., if the quality of the hierarchy is not very low), it can be beneficial to control the FDR with the simple BH procedure.

3.8.3 Real Public Health Data Set

In this experiment we validate our approach against a real-world public health data set. We show that there exist real data sets with the set of properties we describe in this paper that can lead to a gain in statistical power. Our running experiments throughout this paper have been based on this data set [80]. Scientists collected 169 *factor* variables from a questionnaire about the level of physical activity of older individuals. They also collected 36 *outcome* variables about various health-related metrics (such as blood pressure).

It is interesting to note here a methodological difference between the statistics and the database field. In statistics, the prevalent experimental methodology when studying a new technique is to create synthetic data according to a known probability distribution. Then the statistics researchers investigate the ability of their algorithms to correctly uncover hypotheses about the hidden from the algorithm probability distribution, in different settings [12]. In contrast, the database community prefers to read meaningful real-world examples. The problem with this mindset is that the hidden probability distribution is never known in practise. In real-world examples, only a finite sample is known: in the example, 350 individuals with their factor and outcome values. Given an infinitely large sample, the distribution would be fully revealed.

In the interest of bridging the two mindsets, we follow the technique of [88] in determining a plausible distribution: first, we treat our sample of 350 individuals as a good enough approximation of the underlying probability distributions. To determine the underlying distribution, we found the significant correlations in this population while controlling with the Bonferonni correction, the most conservative approach of controlling multiple hypotheses. We then used only a random s part of the initial sample as our “experiment sample” to introduce

randomness. We performed our experiments for various values of s . Also, our ground truth is likely to be biased towards approaches that are similar to Bonferonni.

Figure 3.6 shows the results in the real public health data set for different sample sizes and for two different controlling techniques. We see similar results for both controlling techniques. First, we see that we achieve up to 1.8x improvement in statistical power. We see that as the sample size increases, all approaches achieve higher power. This is because a bigger sample size means less randomness and, as a result, it is easier to identify the discoveries that exist in the data. We also see that we have a smaller gain in statistical power compared to most cases of the simulations. This can be due to multiple reasons. First, as the initial population is quite small, we cannot introduce a high degree of randomness and still have a big enough sample size. Second, this data set has only one level in the hierarchy and it describes the factor variables only.

3.8.4 Data driven Hierarchy

In this experiment we investigate the effect of data driven hierarchies on the gain of statistical power of VigilaDE's data exploration algorithms. We have already discussed that data-driven groups have interpretability problems that make them a poor choice for applications and, more importantly, they are limiting to the kinds of input variables. As a result, it is meaningless to investigate the effect of a data driven hierarchy in the application scenario of this work. We run our experiments for the synthetic data of our simulations.

We used k-means clustering algorithm to create a data-driven hierarchy for factors and outcomes. We set $k = 100$, which is the same number of groups that we have in the a priori hierarchy we create. As k-means clustering creates groups of different sizes, we could not create a hierarchy with more than one level of grouping. In order to make a fair comparison, we run our algorithms with an a priori hierarchy that had only one level of grouping as well. We controlled false discoveries with the GBH procedure in all cases.

Figure 3.5 shows the results with the synthetic data for a data-driven hierarchy and the

input a-priori hierarchy. The performance of the data-driven approach is independent of the quality of the hierarchy that appears on the plots. The performance of the data-driven approaches is the same across all four plots in Figure 3.5. In the Figure we see that the a priori hierarchies outperform the data-driven ones when the quality of the a priori hierarchy is high. As the quality gets lower, then the data-driven approach starts to perform best. For specific applications, where the limitations of data-driven hierarchies explained earlier are not important, creating data-driven hierarchies for our proposed data exploration techniques is an alternative solution. It can even be a preferred solution when we are not confident in the quality of the input hierarchy.

3.8.5 Isolating Effects on Power

In this section, we isolate the different conditions that affect the gain in statistical power. We motivated these conditions in Section 3.7 and here we validate them experimentally. We show that a higher sample size always results in better results, when all other conditions are equal. In addition, we show how the effect of a factor on the outcome is related to power and how this effect appears in hierarchical data.

Sample size. We examine how different sample sizes affect statistical power when we have hierarchical data. As we cannot explicitly vary the sample sizes, we vary the number of variables that belong to a group, and as a result the sample size of the group variable against the child variables. We created pairs of $\langle factor, outcome \rangle$ for 10 outcomes and 20 group factor variables. We measured power for different ratios of sample size of group/AVG(sample size of children). A ratio of 2 means that the group factor has 2 child factor variables and as a result double the sample size. All child factors in a group come from the same population. In each scenario, we tested all child factors and all outcome pairs against all group factors with all outcome pairs. We see the results in Figure 3.7(a),(b). There is a higher gain in statistical power when the group variables have a larger sample size (higher ratio), whether we control for the FDR or not.

Effect on outcome. We examine how the effect on the outcome affects power in hierarchical data. In this experiment we validate our intuition of equation (2). We measure power for different cases of the quality of a hierarchy, a metric that measures the percentage of the child variables in a group that come from the same distribution. We created pairs of $\langle factor, outcome \rangle$ for 10 outcomes and 20 group factor variables. We measured the statistical power for different qualities of the hierarchy. A quality of 90% means that in each group, 90% of child factors come from the same distribution. In each scenario, we tested all child factors and all outcome pairs against all group factors with all outcome pairs. We see the results in Figure 3.7(c),(d). When we do not control for false discoveries, there is a certain threshold where it stops being beneficial to test the group variables. Interestingly, this is not the case when we control the FDR. We can still see that as quality lowers we have a smaller gain in power. However, in all cases it is beneficial to test the group variables. In this scenario, controlling lowers the power of testing the children enough (to around 0.25) that even when the quality of the hierarchy is low the power of testing the group variables doesn't become lower than that.

3.9 Related Work

In Section 2.2 and 3.5 we examined a lot of related work from statistics about the problem of controlling false discoveries. Next, we will discuss recent work about data exploration systems and bayesian models.

Data Exploration. A lot of work recently in the data management community focuses on different data exploration systems and techniques [49, 74, 77, 82, 85]. Idreos et al provide an overview of recent data exploration systems [44]. OLAP graphical tools utilize hierarchical data to provide drilling and slicing; Polaris is a common example [77]. Sellam et al focus on clustering data as a way of providing an overview in the exploratory process [71]. Bernard

et al describe a system that provides visual summaries and drilling of time series data [13]. In contrast to our work, these systems create data-driven clusters with the goal of providing a summarization of results and improve the user interaction. However, they do not take into account false discoveries, and as data driven approaches to creating a hierarchy they still have limitations for our scenario as we discuss in this paper.

Some systems allow hierarchical data, described by an ontology, to be visualized by a graph and drill down to explore certain parts of the ontology [10]. However, these system focus only on improving the user interaction.

The authors of [88] proposed new α -investing rules to control false discoveries in interactive data exploration. During interactive data exploration hypotheses appear in an adaptive and streaming way. As a result, most previous techniques could not be applied. α -investing is a sequential technique. The authors come up with a few novel investing techniques in order to achieve the best results in different interactive data exploration scenarios. If a_i invested is too high, the entire wealth might be exhausted too fast and the procedure might need to stop. On the other hand, if a_i is too low, it can result in losing a lot of statistical power.

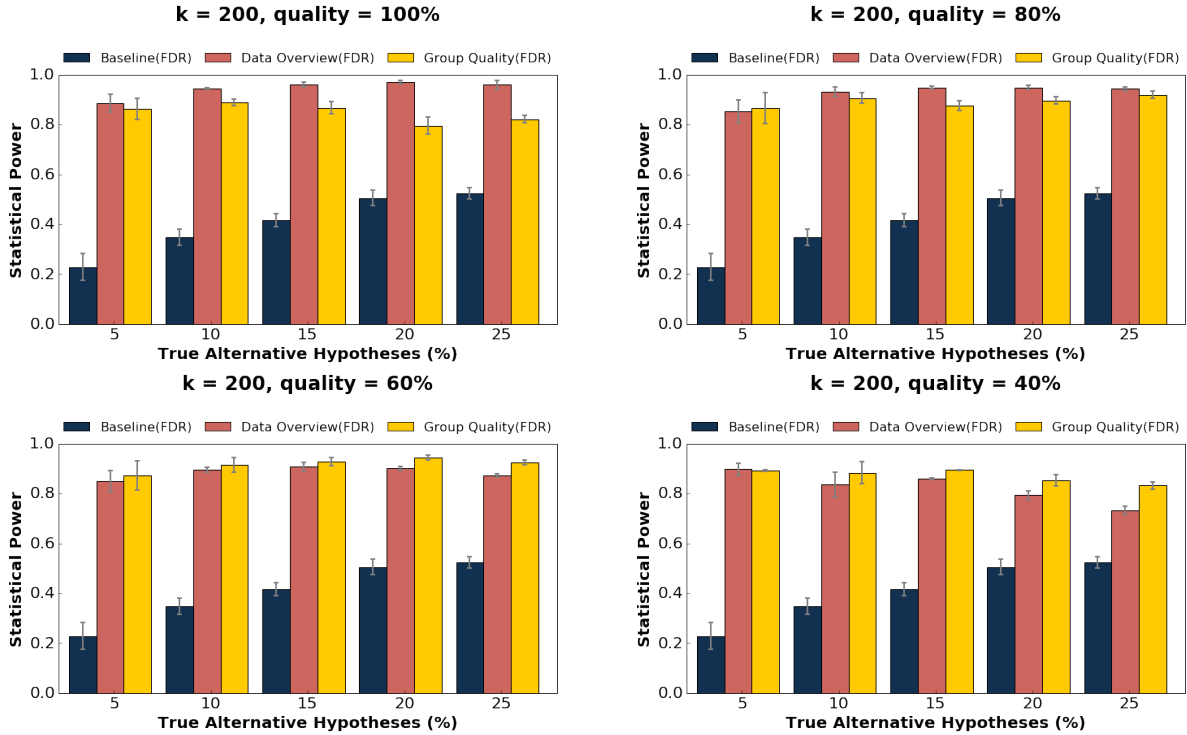
Bayesian models. Bayesian models are an alternative to frequentist statistical methods for inference [34]. One advantage of Bayesian models is that users usually do not have to worry about false positives that come from multiple hypothesis testing [32]. However, these models require additional effort in the form of modeling prior and posterior probabilities in order to capture specific field knowledge. When incorporating such field knowledge in prior probabilities, these models have been shown to perform very well [33]. The manual effort of selecting priors can be alleviated by automated methods, but they still require some kind of field expertise [86]. VigilaDE, as a data exploration system, does not require users to have any kind of specific field expertise. As a results, Bayesian methods for controlling false discoveries can be unsuitable in our setting.

3.10 Conclusion

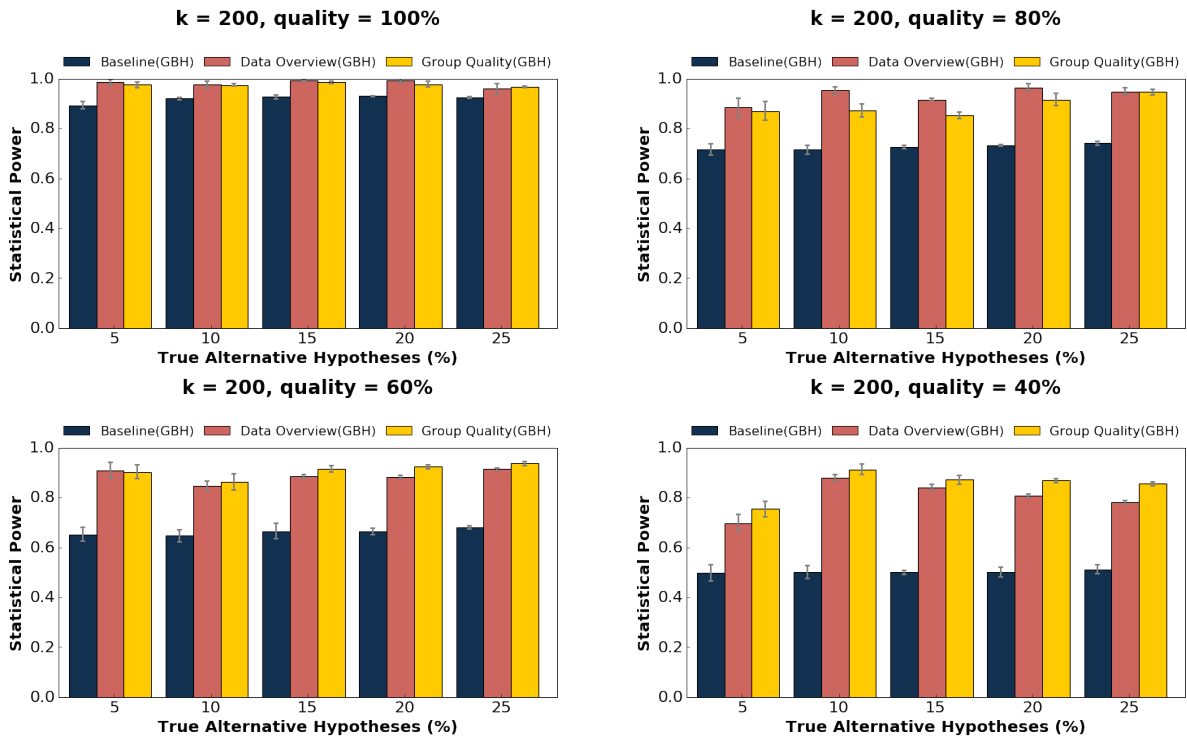
In this chapter, we presented a novel way to explore hierarchical data that can increase statistical power when we control for false discoveries. Our analysis and experimental evaluation showed that we can achieve a significant increase in statistical power against existing controlling techniques.

Acknowledgements

This chapter contains material adapted from the paper “VigilaDE: Avoiding False Discoveries with Hierarchical Data in Data Exploration Systems” by Nikolaos (Nikos) Koulouris; Arun Kumar; Yannis Papakonstantinou. It is currently under review for publication. The dissertation author was the primary investigator and author of this paper.



(a) Controlling the FDR with Benjamini-Hochberg procedure.



(a) Controlling the FDR with Group Benjamini-Hochberg procedure (GBH).

Figure 3.4. Experiment of different data exploration algorithms for varying parameters of data with two different controlling techniques.

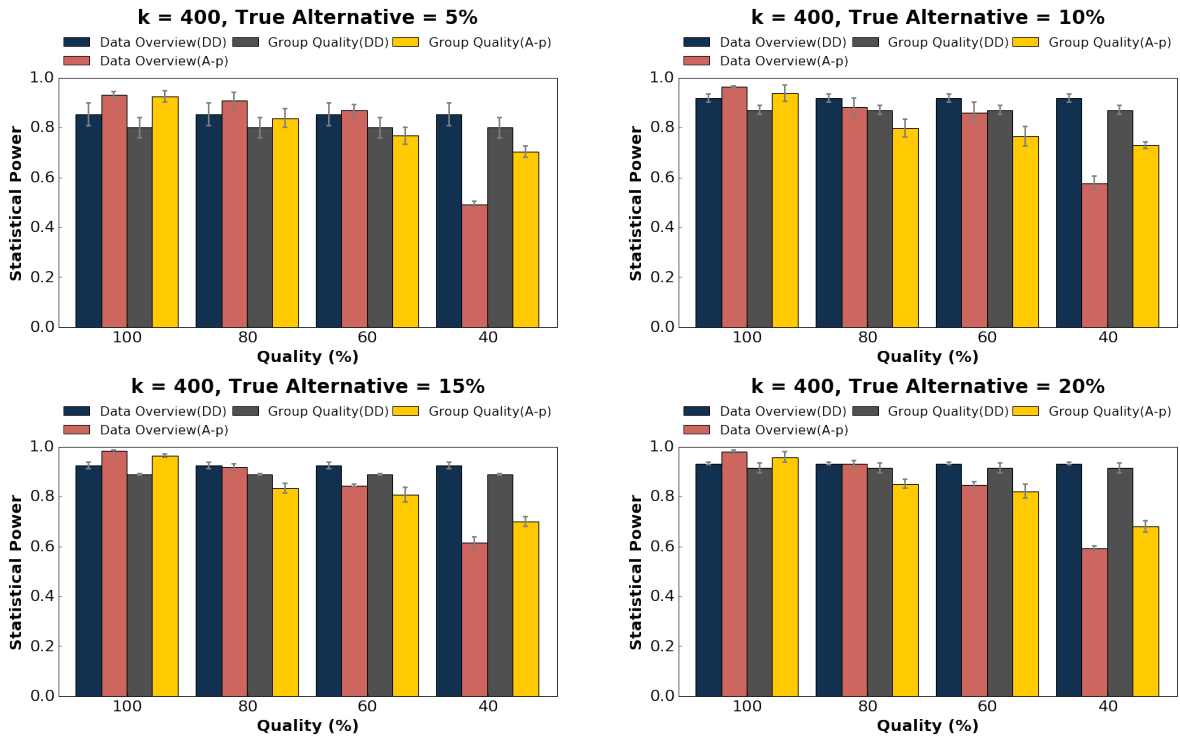


Figure 3.5. Comparison of Statistical power of our algorithms with data-driven(DD) and a priori(A-p) hierarchies. False discoveries were controlled with the Group Benjamini-Hochberg procedure.

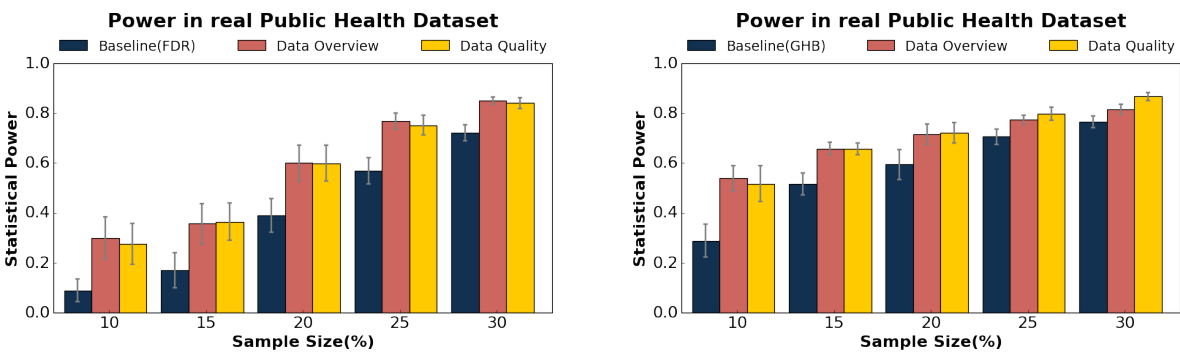
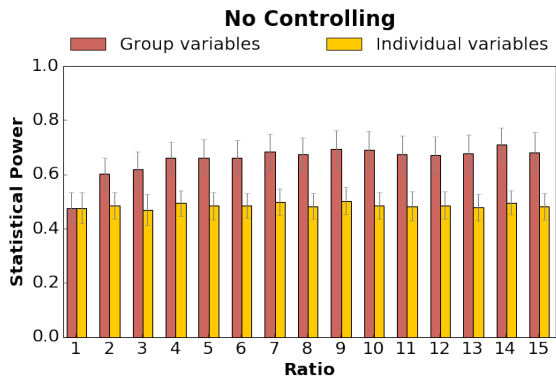
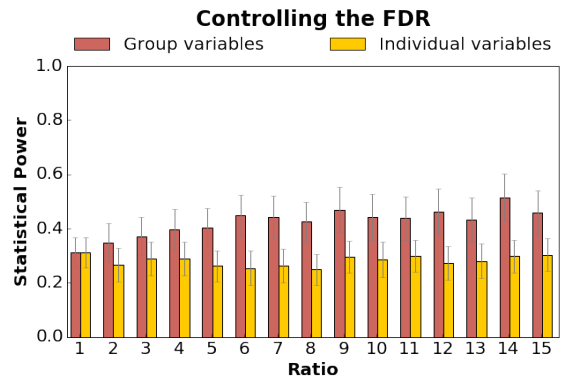


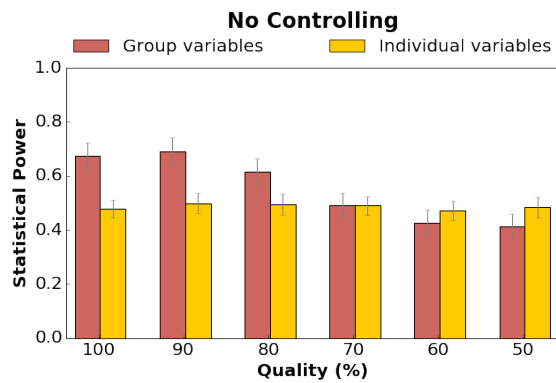
Figure 3.6. Experiment with real Public Health data set.



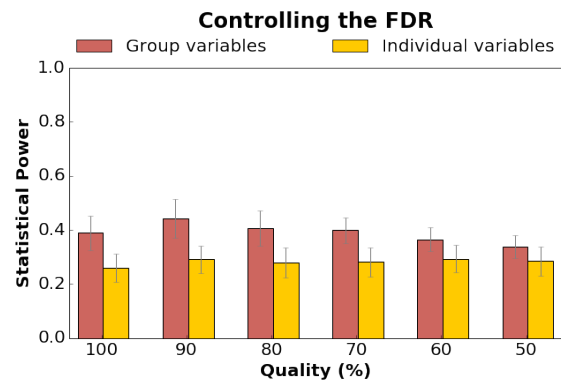
(a) Sample size and power in hierarchical data without controlling.



(b) Sample size and power in hierarchical data when controlling FDR



(c) True quality of a hierarchy and power in hierarchical data without controlling.



(d) True quality of a hierarchy and power in hierarchical data when controlling FDR.

Figure 3.7. Experiment to show how the sample size and the effect of a factor on an outcome affect statistical power.

Chapter 4

Avoiding Validation Overfitting during Feature Selection

In this Chapter, we present a novel algorithm for feature selection that avoids overfitting the holdout validation dataset. First, we present an experimental analysis of the state of the art algorithm for avoiding overfitting the hold out dataset in a generic setting and provide guidelines for setting its parameters. Then we present AUTO-SET, a novel and automated way to set these parameters. Lastly, we present Auto Adjust Threshold, a novel feature selection algorithm that avoids overfitting a holdout dataset and we show that it outperforms existing algorithms.

4.1 Motivating Example & Overview

A user has a dataset consisting of 3000 samples and 3000 features. He wants to build a classifier with a logistic regression to predict a binary outcome. As it is common practise he splits the dataset into a training, holdout/validation and testing set. In the dataset of this example, the output binary variable is just a random variable; i.e. no feature has any predicting power. To avoid overfitting the training data he decides to select only 200 features from the dataset. To achieve that he performs sequential feature selection, where at each step he chooses to add the next feature that is most correlated with the response variable. To validate the feature selection

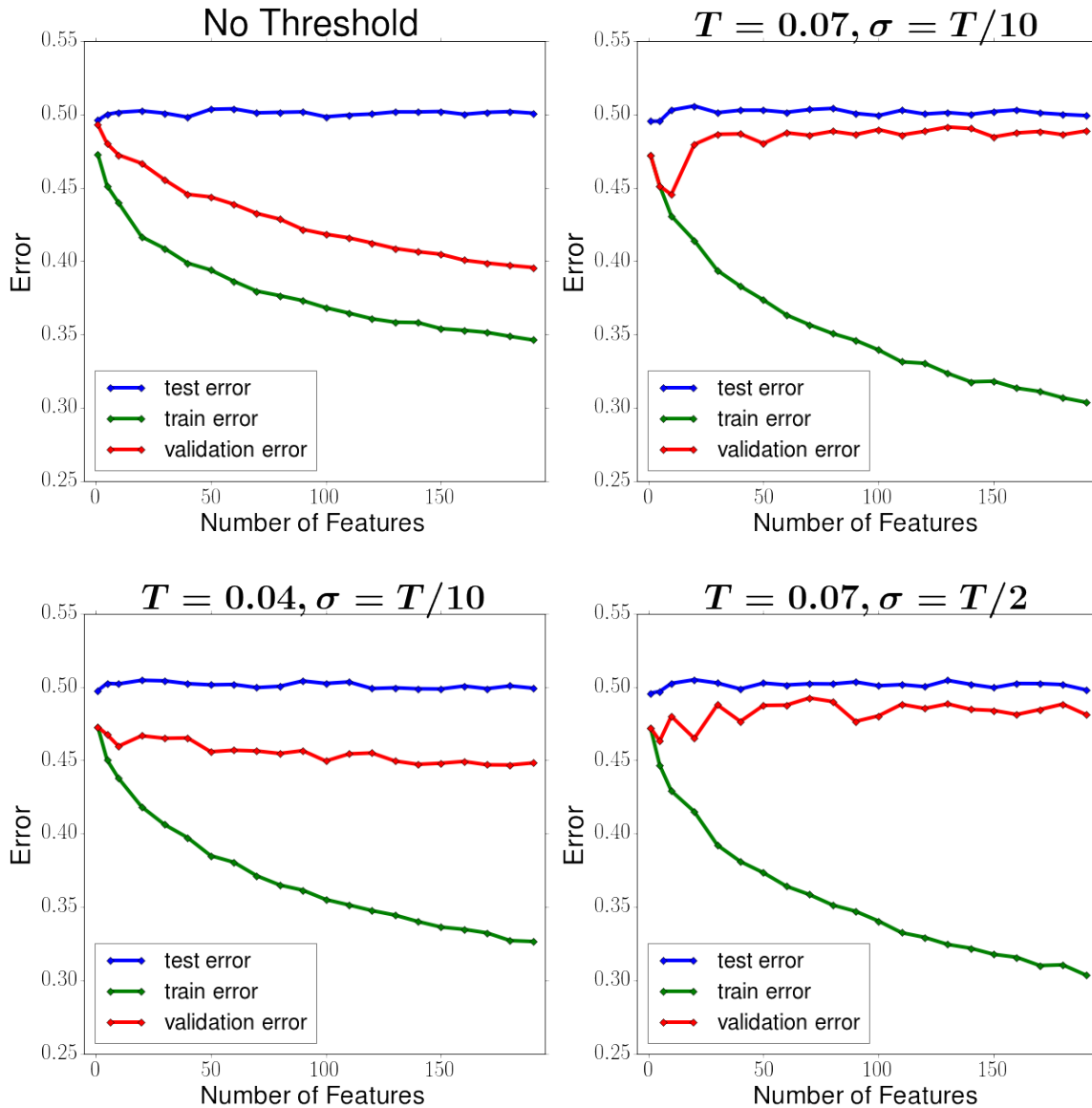


Figure 4.1. Training, validation and test error during feature selection in four different scenarios.

process he uses the holdout set.

In Figure 4.1 we see the error of this classifier in four different scenarios as a function of the number of features that were selected. In the top left plot, which is the scenario we just described, we can clearly see that this ML model overfits the validation set. The validation error curve's trend follows the trend of the training error curve. They are both decreasing as the model has more features. However, the test accuracy, which is not used throughout the feature selection

process, stays constant at 50%. Obviously, the validation set error overfits the data and cannot be used to estimate the generalization error.

In [27] the authors proposed a solution for this problem. Their proposed algorithm ThresholdOut is a way of accessing a dataset. Its idea is simple. If the value of a metric that we want to compute differs more than a predetermined value τ on the training and the holdout datasets, then return to the user the value on the holdout dataset. Otherwise, when the values are close enough the user does not need to know the actual value on the holdout dataset. In formal description of the algorithm there is also some noise σ added. This process of accessing a holdout dataset helps avoid overfitting.

However, to use ThresholdOut one first needs to set its parameters τ and σ which are not very intuitive. In Figure 4.1, we see the results of feature selection with ThresholdOut for three different pair of values for its parameters. In the top right plot, the chosen parameters seem to work very well as we avoid overfitting the validation dataset. When the threshold is too small (bottom left plot), we still have less overfitting than when not using ThresholdOut, but more than in the previous scenario. Lastly, if noise σ is too high (bottom right plot), this can lead to having less stable results which can make our feature selection less stable in converging to the best solution.

Overview. ThresholdOut can be very useful in reducing validation overfitting, however all of its benefit can disappear when the wrong values for its parameters are chosen. In Section 4.3, we show the results of our experimental analysis of how setting these parameters affects the performance of ThresholdOut during feature selection in many different settings. We also provide intuitive guidelines for setting these parameters.

In Section 4.4 we automate the guidelines for setting the parameters of ThresholdOut. We present AUTO-SET, an algorithm that automatically picks the values of the parameters τ and σ depending on the setting. We experimentally show that AUTO-SET in most cases performs very close to the optimally picked values for τ , σ .

In Section 4.5, we present Auto Adjust Threshold(AAT). AAT is a novel approach to feature selection that is based on the main idea of ThresholdOut and avoids overfitting the holdout dataset. Feature selection is an adaptive process where the “queries” that we want to evaluate on the dataset have a very specific structure. For example, in forward selection we start from a very simple model and adaptively test more complex ones. AAT exploits this structure to automatically adjust the value of τ in each step of the feature selection algorithm in order to reduce overfitting and, as a consequence, reduce the error.

4.2 Background

In this section we explain intuitively the ML terms and concepts we use throughout this paper and refer readers to the related textbooks for a more in-depth background [37, 58, 73]. We also give a brief description of ThresholdOut algorithm [27].

4.2.1 ML Concepts

This work focuses on supervised learning. Supervised learning involves learning a machine learning (ML) model from a training dataset of correctly labeled examples. The training dataset is used to learn the parameters of the ML model and then the ML model can be used to make predictions for new unseen examples. There are many different ML models for different settings and problems. For example some ML models that we will use for classification tasks are logistic regression, decision trees, Naive Bayes classifiers. To measure a model’s performance and compare it against other models, we use the test error, the error of the model on a holdout test dataset of previously unseen examples.

Feature Selection. Feature selection (FS) is the process of selecting only a subset of features (variables) to use in building the ML model. FS methods are almost always used along with a ML model to help improve accuracy, among other reasons. At a high level there are three types of feature selection methods: wrappers, filters and embedded methods [35]. In this work we focus on *wrapper* methods.

Wrapper methods use a search algorithm to search the space of possible feature subsets to obtain the most accurate one. Wrappers use the ML model as a “black-box” to score all these different feature subsets. Sequential greedy search is one of the most popular search algorithms for wrappers. It has two variants: forward stepwise selection and backward stepwise selection. Given a feature set \mathbf{X} , forward (resp. backward) stepwise selection adds (resp. deletes) the best (resp. worst) feature at each step, starting with the empty set (resp. full set). To determine the best (resp. worst) feature at each step it computes the error of an ML model with (resp. without) the candidate feature. The error can be measured using a holdout dataset or using k-fold cross-validation. For our purposes, we use the simpler holdout method commonly used [37]: the input data is split 50% : 25% : 25% with the first part used for training, the second part used for the validation error during greedy search, and the last part used for the holdout test error.

Overfitting. When a ML model describes a limited set of data points too closely, then this model *overfits* the data and can fail to generalize to new unseen examples. This is a common problem in ML that can appear easily when the ML model is too complex (i.e. decision trees) or when the amount of data points is limited. Most of the existing techniques focus on fixing the problem of overfitting the training data. There are many other more subtle ways to overfit data, such as overfitting the validation dataset during procedures like feature selection [53].

Model complexity. Some ML models are simpler than others. For example, always predicting the average value of the training data is a much simpler than fitting a linear regression model which is simpler than a decision tree model. One common way to measure the complexity of a ML model is its VC dimension.

Intuitively, the VC dimension measures the maximum number of any data points that the model can correctly classify; a capability known as shattering. For example, consider a binary linear classifier in 2-D. It is easy to see that this model can correctly classify any set of 3 points (distinct and noncollinear). But, it cannot shatter any set of 4 points. Thus, its VC dimension is 3. The definition of VC dimension can be extended to regression ML models [37].

A higher VC dimension means probably a higher variance. The VC dimension *usually* increases with the number of features. For example, it is linear in the number of features for linear classifiers, such as logistic regression and Naive Bayes [61]. As a result, feature selection usually produces simpler ML models.

Bias-variance trade-off. The test error can be decomposed into three components: bias (a.k.a approximation error), variance (a.k.a. estimation error), and noise (a.k.a. irreducible error). The noise is an inevitable component that is independent of the ML model. Bias is the error that results from erroneous assumptions about the model. It measures how far is the best learner in the model from the optimal one. Variance is the error that results from the the fact that the training dataset is only a random finite sample from the underlying data distribution. It measures how sensitive the model is in fluctuations in the training data.

Model complexity can be used to explain bias and variance. A model of low complexity introduces some kind of assumption (bias) about the underlying data distribution. For example, using a linear classifier assumes the data can be described with a linear model. A model of high complexity can have high variance as it might “memorize” (overfit) the data rather than learn the underlying distribution. It is obvious that there is a *trade-off between bias and variance* as we go from simple to more complex ML models.

The feature selection process tries to improve performance by finding the feature set that minimizes both bias and variance. We go more in depth about how this trade-off appears during feature selection in later sections.

4.2.2 ThresholdOut

ThresholdOut [27] is a method for reusing a holdout dataset to validate *adaptively* chosen hypotheses while avoiding overfitting the holdout set. This method also provides some generalization guarantees. The main idea is the following: the user can form a hypothesis (i.e., the best features for a ML model) with access to the training data. But, to access the holdout data for validation he has to use the ThresholdOut method. This method checks whether the hypothesis overfits the training data by comparing it with the holdout data. It returns the mean value of the hypothesis on the holdout data *only if* the hypothesis overfits the training data. In this way, the minimum information about the holdout data, that is needed to avoid overfitting the training set, is leaked to the user. As a result, ThresholdOut also avoids overfitting the holdout data.

In Algorithm 1 we see the pseudocode of Thresholdout as it is presented in [27].

ALGORITHM 1: ThresholdOut pseudocode

```
Input : Training set  $S_t$ , holdout set  $S_h$ , threshold  $\tau$ , noise  $\sigma$ , budget  $B$ , multiple hypotheses  $\phi_i$   
1 while  $B > 0$  do  
2   if  $|E[\phi_i(S_t)] - E[\phi_i(S_h)]| > \tau + Lap(8\sigma)$  then  
3     output:  $E[\phi_i(S_h)] + Lap(\sigma)$   
4      $B = B - 1$   
5   else  
6     output:  $E[\phi_i(S_t)]$   
7   end  
8 end
```

S_t and S_h are the training and holdout datasets respectively. τ, σ are two input parameters for the method. The threshold τ determines what difference between the training and holdout average values is considered overfitting. The noise parameter σ , also called tolerance, determines how much Laplacian noise ($Lap()$) to be added to the results and it affects the formal guarantees given by the authors. It is also called tolerance, because it determines how much “off” can be the average value of the hypothesis on the holdout set that the algorithm returns. ϕ_i is i -th adaptively chosen hypothesis that the method takes as input. Lastly, the budget B ,

is determines how many queries the method can answer about different hypotheses without overfitting the holdout data.

The main idea of the pseudocode is simple. If the difference of the average score of ϕ_i on the training set and the holdout set is bigger than the threshold τ and some noise σ , then ϕ_i overfits the training data, so the method returns the score on the holdout data. Otherwise, the user doesn't need to know the score of ϕ_i on the holdout data.

4.3 Analysis of ThresholdOut

In this section we examine how different values of the parameters of ThresholdOut affect overfitting and test error. First, we observe the performance of ThresholdOut for different values of τ and σ on a synthetic dataset, for different ML algorithms. Next, we validate these insights on real datasets. In the next sections, we present these insights for wrapper feature selection methods. We conclude by giving generic guidelines for setting these parameters.

4.3.1 Design of Experimental Analysis

We evaluated the performance of ThresholdOut on many different settings. We examined the ML problem of feature selection for binary classification. We tested wrapper methods, specifically stepwise wrapper methods. We trained many different ML models, a Naive Bayes model, a Logistic Regression model and a Decision Tree model. We performed our analysis for synthetic datasets as well as real ones. For each different setting (ML algorithm, dataset, feature selection algorithm), we varied the parameters threshold $\tau \in [0.01 - 0.2]$ and noise $\sigma \in [0.001 - 0.1]$. More details about our experimental setup can be found in Section 3.8.

To evaluate ThresholdOut, we reported the test error e_t of the model when the feature selection algorithm converged and the overfitting there. We measure overfitting as the difference

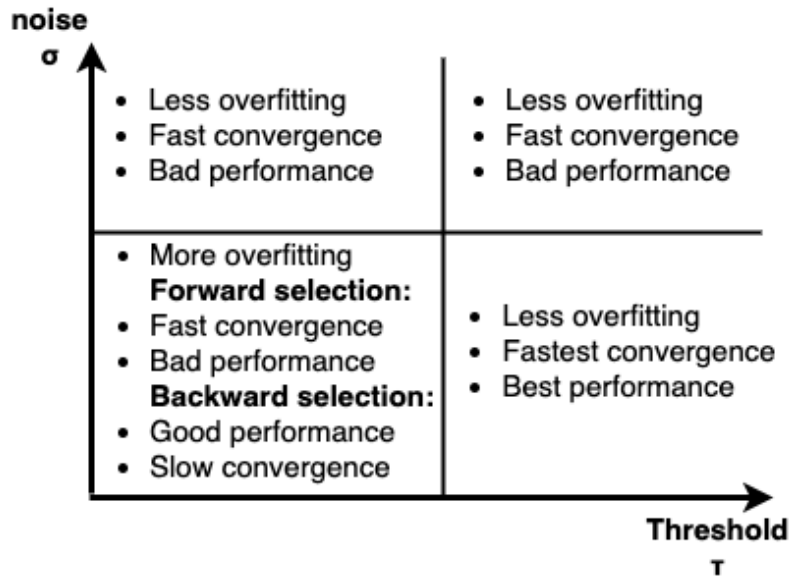


Figure 4.2. Summary of results of analysis of ThresholdOut algorithm in Forward and Backward Selection.

between the test error and the validation error $|e_v - e_t|$. To get a better picture of overfitting we also reported the overall overfitting of the process, as the average overfitting at each step of feature selection.

Next, we present our insights of this extended experimental analysis. We present these results separately for forward and backward stepwise feature selection because even though the high level insights are almost the same, the reasons for these results are different. The full results of the experiments can be found in Section Section 3.8.

4.3.2 Forward Selection

We varied the threshold τ and noise σ parameters of ThresholdOut algorithm and observed how they affected the results of forward selection. Because there are so many different possible settings, we give some insights in relative terms, such as better or worse performance, based on our observations from our experimental analysis. The experimental results are summarized in Table 4.3 through Table 4.7 and our insights in Figure 4.2.

When τ and σ are low, we have the most overfitting. This is very similar to feature selection without ThresholdOut. A small threshold allows the validation error to decrease faster than the actual error and feature selection overfits to the validation set. This results in converging faster to a solution as validation error decreases faster. Another result of this overfitting is that the final performance on the test set is usually worse.

For higher values of τ and still low values of σ , we have less overfitting, both overall during feature selection, and at the final solution. Convergence to the final solution is still fast, and even faster than with small threshold τ values as the minimum validation error reached in this case is not as small as when there is more overfitting. Lastly, lower overfitting results in better performance than before.

For low values of τ and high values of σ , ThresholdOut has a lot more noise. First, σ determines the noise added to the threshold τ . As a result, even if τ is small, a high σ value can lead to sometimes having a high threshold. This results in less overfitting mostly due to randomness introduced by a high σ . Moreover, σ determines the noise added to the validation error that ThresholdOut returns. This can affect the feature that the algorithm picks at each step when there is a lot of noise and, as a result, there is a more gradual decrease in the validation after every step.

4.3.3 Backward Selection

Next, we repeated the previous experiments for backward selection. These results are summarized in Table 4.3 through Table 4.7 and our conclusions are summarized in Figure 4.2.

When τ and σ are low, we have the most overfitting in backwards Selection as well. A small threshold value allows to overfit the validation set and in backwards Feature selection that means that convergence can be slow as the algorithm starts with a very low validation error and stays that way longer. Slow convergence in backward selection means fewer features, which is a positive. Lastly, the final performance on the test is good.

For higher values of τ and still low values of σ , we have obviously lower the overfitting. In backward Selection this doesn't seem to affect the convergence speed and with the right τ value it can even improve the final performance on the test set.

For low values of τ and high values of σ , we have more noise. As we explained before, this means the threshold τ will be sometimes larger and thus we have less overfitting. However, noise changes the convergence speed. With more noise the validation error slowly increases and as a result the feature selection algorithm stops fast. As a result, the algorithm can return too many features and overall its performance is worse.

Lastly, for high values of both τ and σ , we have a performance that is similar to low τ and high σ . The difference is that the big threshold value results in a bit slower convergence speed.

4.3.4 Guidelines for setting τ , σ

Based on the insights we described about τ and σ , we give some guidelines for setting these parameters for wrapper feature selection algorithms.

- The noise parameter σ should always be an order of magnitude smaller than the threshold parameter τ . Since noise is also added to the threshold τ itself, having a bigger σ than τ results in a a lot of random variability in the threshold at each step.
- The noise parameter σ should be small. Large σ values can make the feature selection algorithm unstable; it might pick the wrong features at each step because noise is added to the validation error that ThresholdOut algorithm returns. As a result, large σ values can make the feature selection algorithm converge to a solution that doesn't perform that well. A good rule of thumb based on our experiments is that σ should be at most $\tau/10$.
- The threshold parameter τ should be large in order to prevent overfitting to the validation set. A too small value will lead to a lot of validation overfitting. On the other hand, a too

large value can have bad performance due to training overfitting.

How much smaller than τ should σ be? How small should σ be? And how large should τ be to avoid overfitting both the validation and the training datasets? The exact values for σ and τ that will give the best results *depend on the problem setting*. As a result, our guidelines are generic and not very practical. We cannot quantify them without knowing the specific dataset, ML algorithm and feature selection process. A threshold of $\tau = 0.1$ might give the best performance for one dataset, but might be too large and lead to training overfitting on another dataset. To solve this problem in the next section we present an algorithm to automatically select values for τ and σ based on the problem setting.

4.4 AUTO-SET

In this section we start by motivating the need for an automated way of picking the parameters of ThresholdOut algorithm. Next, we present AUTO-SET, our proposed way of automatically setting the parameters τ and σ for ThresholdOut algorithm for wrapper methods of feature selection.

Motivation. In the previous section we gave some generic guidelines that help users set the parameters of ThresholdOut, but they were generic because the optimal values depend on the problem setting. For example, in our analysis we saw that on the synthetic data, for logistic regression, for forward selection $\tau = 0.20$, $\sigma = 0.001$ were the best performing values in regards to the error among all the possible parameters we tried. In contrast, for the same setting but for backward selection the best parameters were $\tau = 0.05$, $\sigma = 0.001$. Backward selection performed better for a relatively small value of τ whereas forward selection performed best for the largest value of τ we tried. This example makes it obvious that there is not one way to set these parameters to have the best performance across all different problem settings.

We need to take into account the feature selection algorithm, the ML model and some initial information about the data in order to select good values for τ and σ . In addition, as we have already discussed, autoML systems make solving this problem a necessity. By their definition they have to automate ML processes because their users are not ML experts.

Intuition. Our intuition of how AUTO-SET picks the values for ThresholdOut algorithm come from the bias-variance trade-off as it is illustrated on Fig 4.3. We use the bias-variance trade-off to explain the different training and validation error we expect to see in feature selection. The trade-off is not the same for forward selection, where we start with the simplest ML model, and for backward selection, where we start with a very complex ML model. We explain how we can leverage the bias-variance trade-off in these two different situations to pick values for the parameters of ThresholdOut.

Forward selection algorithms start with an ML model with only one features (a very simple model) and adds more features at each step until it can't reduce the validation error of the model any further. That means that FS algorithms start with a model with low variance and high bias. Low variance means that the ML model does not overfit the training data. As we see in Figure 4.3 as well, the training error(e_t) is a *good* approximation of the true error. This also means that we do not need to use the validation error during the first steps of feature selection.

Backward selection algorithms start with an ML model with all the input features (a complex model) and remove features at each step until it can't reduce the validation error of the model any further. That means that backward selection starts with a model with high variance and low bias. High variance means that a complex ML model can easily overfit the training data. This means the training error is not a reliable approximation of how well the model generalizes in new unseen data. This can be seen in Figure 4.3, as the difference between the training and the true error when a model is complex. As a result, when our model is too complex using the training error to guide the feature selection is not helpful, and the validation data is necessary.

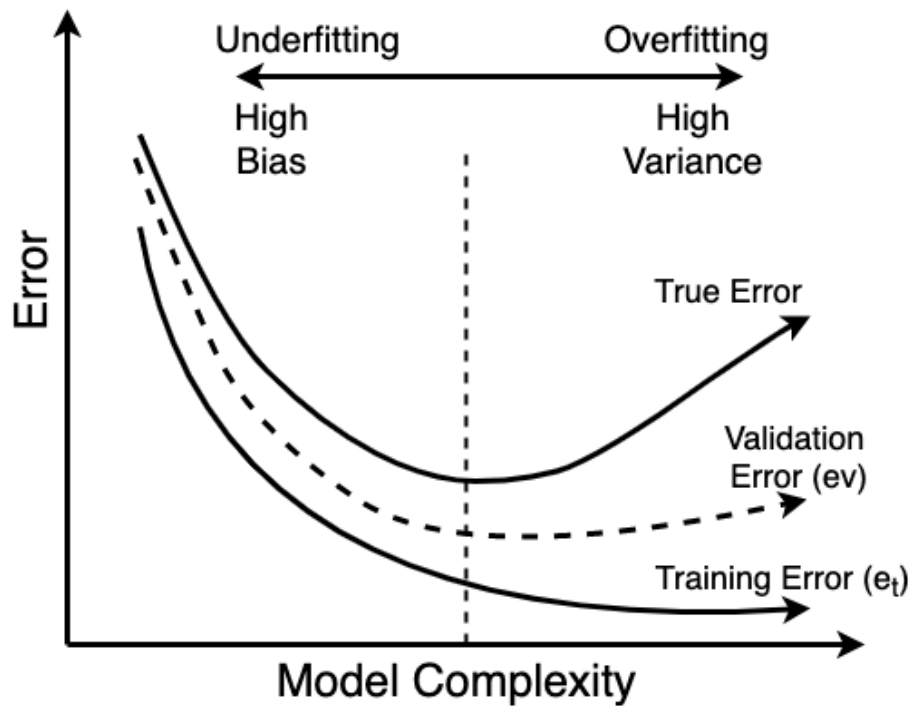


Figure 4.3. Bias-variance trade-off.

4.4.1 Forward Selection

Simple ML models have the low variance. At the first steps of forward selection the model is simple and does not overfit the training data. Also, due to low variance the training and validation errors should be very close. AUTO-SET picks for τ (almost) the biggest difference between the training error e_t and the validation error e_v on the initial step of forward selection. By doing this, ThresholdOut rarely returns the validation error, and forward selection picks which feature to add based mostly on the training error. As the model gets more complex, the variance increases, the model starts to overfit the training data, and there is a bigger difference between e_t and e_v . As this difference gets bigger, ThresholdOut returns the validation error e_v more often.

Setting τ . AUTO-SET computes the differences between the training error e_t and the validation error e_v of all the possible initial feature sets FS_1 .

$$diff = \{|e_v - e_t| \mid \forall FS_1\}$$

And then it picks as τ the value:

$$\tau = \max(\{x_i \in diff \mid x_i < \bar{x} + 3\sigma_x\})$$

where \bar{x} is the mean of the *diff* set and σ_x its standard deviation. We do not set as τ the maximum value in *diff* in order to avoid outlier values [].

Setting σ . From our analysis we saw that σ should be much smaller, and preferably an order of magnitude smaller than τ . This makes intuitive sense as well, as noise is also added to τ based on σ value. So picking a $\sigma > \tau$ results in a highly variable threshold τ . On the other hand, a very small value of σ does not help prevent overfitting the validation data. From our analysis we saw that picking for $\sigma = \tau/15$ is a good rule of thumb that gives good results. In all our methods we adopt this way of setting σ . When we do not mention how σ was set, we imply that we used this method.

4.4.2 Backward Selection

Backward Selection works in the opposite way of forward selection. It starts with a complex model (i.e., a model with all the input features). Complex models have a high variance. High variance means that the model can overfit the training data and results in a big difference between e_v and e_t . AUTO-SET picks for τ the mean value of the differences between the training error e_t and the validation error e_v on the initial step of backward selection. By doing this, backward selection picks which features to remove based both on the training set and the

validation set on the first steps. As the model gets simpler, the variance decreases, the model overfits less the training data, and the difference between e_v and e_t becomes smaller. As this difference gets smaller, ThresholdOut returns the training error e_t more often.

Setting τ . AUTO-SET computes the differences between the training error e_t and the validation error e_v of all the possible initial feature sets it tries FS_k , where k is the number of all input features.

$$diff = \{|e_v - e_t| \mid \forall FS_k\}$$

And then it picks as τ the value:

$$\tau = \bar{x}$$

where \bar{x} is the mean of the *diff*.

We presented AUTO-SET, our method for automatically picking the values for the parameters τ and σ for ThesholdOut algorithm for wrapper methods of feature selection. The bias-variance trade-off gives us more information during the feature selection process that we can use to further optimize our method. As the model complexity changes during feature selection, the initially picked threshold τ is not a good choice anymore. We use this information to further improve upon ThresholdOut.

4.5 Auto-Adjust Threshold feature selection

In this section we present Auto Adjust Threshold(AAT). AAT is a novel method of accessing the holdout validation dataset during feature selection with wrapper methods. AAT allows reuse of the validation data during feature selection only when necessary to avoid overfitting the validation data and, as a result, improve model performance. We start by motivating the opportunity to exploit the structure of the feature selection process to improves the final model

performance. Then, we give an overview of our procedure before we dive into the technical parts.

4.5.1 AAT Overview

Auto Adjust Threshold is a novel approach to accessing a holdout dataset for validation during feature selection with wrapper methods. AAT is based on the main idea of ThresholdOut that access to the validation data is not needed to evaluate all models, but takes it one step further. The models that are evaluated during feature selection have a structure when it come to their complexity. Models get more complex during forward selection or less complex during backward selection. As a result, as the complexity of the models changes, the threshold τ that ThresholdOut initially picks is not appropriate anymore. AAT approximates the variance-bias trade-off that is the result of the gradual change in model complexity. It then uses it to adjust the threshold τ .

An example of the opportunity for improvement is the last step of forward selection when we execute forward selection until we have all possible features in the model. This can also be seen as the first step of backward selection. Both have to pick the best among the feature sets FS_k . We previously mentioned that forward selection needs a relatively large τ in its first steps and backward selection needs a relatively small τ in its first steps. It is obvious that the large τ that AUTO-SET picks on the first step of forward selection is not ideal at the end of that feature selection algorithm, or in general, in later steps of the algorithm.

Auto Adjust Threshold uses the bias-variance trade-off to gradually adjust the initial threshold throughout the feature selection process of any wrapper method. An overview of the procedure can be seen in Figure 4.4. To know how much to adjust the threshold, we would need to know ideally the point when the test error is minimum k_{min} . Because we cannot know the test error curve, we approximate it with the holdout validation error curve. The error curve which exhibits a characteristic U-shaped curve based on the bias-variance trade-off [58]. However,

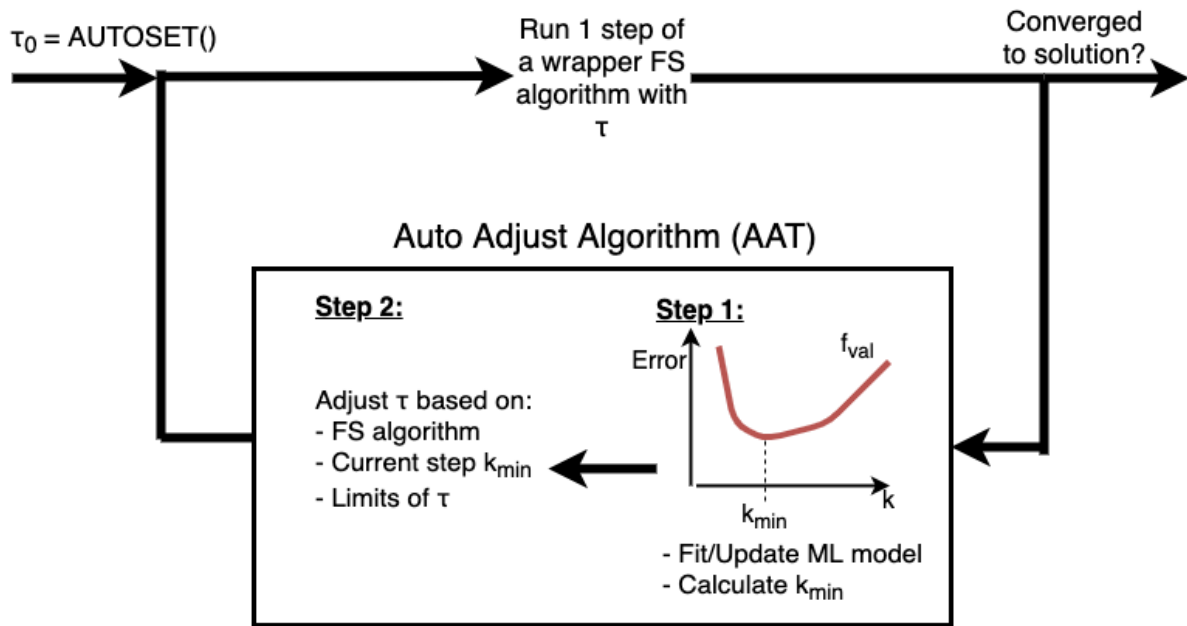


Figure 4.4. Overview of Auto Adjust Threshold feature selection process.

depending on the setting of the feature selection procedure, there can be a big variation in how the error changes as the number of features changes. As a first step, Auto Adjust Threshold builds a machine learning model f_{val} during the feature selection process to approximate this curve. AAT uses the model to predict the minimum point k_{min} of the error curve. In Section 4.5.2 we describe the details of how we build and update this model. As a next step, we use k_{min} along with the minimum and maximum values of the threshold to adapt the threshold τ for the next step of the feature selection algorithm. In Sections 4.5.3 - 4.5.5 we describe how we adapt τ based on different wrapper feature selection algorithms.

4.5.2 Approximating validation error curve

In this section, we describe how AAT builds an ML model during the feature selection process to estimate the validation error curve.

The validation error curve during feature selection has a U-shape because of the bias-variance trade-off [58]. However, on different settings this U-shape form can be significantly

different. For example, decision trees have very high variance but low model bias. As a result, the validation error curve can be much steeper and it can decrease much faster. The minimum point (or the bend point) k_{min} of the curve will be reached much faster than in other cases. The ML task here is to predict the bend point of the validation curve k_{min}^* .

ML model. We build an ML model for the validation error curve during the actual feature selection process. Because we know that the distribution function for the validation error curve is U-shaped we use polynomial regression with second degree polynomials. For our setting we need to be able to update our model online, as more data becomes available at each step i of the feature selection algorithm. At each step i we get a new training point (x_{new}, y_{new}) from the feature set FS_i and we update our ML model. To achieve this we used a stochastic gradient descent optimizer which can update the regression model as more data becomes available.

In Algorithm 2 we see the pseudocode of the procedure of building and updating this model. Our model starts only with the first and last points of the curve. We cannot update the threshold τ until after the first iteration of the procedure, when we have at least three points. At each step i of the feature selection algorithm we get a new training point (x_{new}, y_{new}) . We use this to update the ML model. The updated model is used in the next step to recompute the predicted minimum point k_{min}^* which is necessary to adjust the threshold τ_i for the next step of the feature selection algorithm.

Initial training data. To approximate the U-shaped curve we use a polynomial of degree 2. To approximate such a polynomial we need at least 3 data points. The first point (x_0, y_0) can be computed from the model with all the features (feature set FS_k) independently of the feature selection algorithm, as all algorithms will have the same feature set FS_k . x_0 is the number of features in that model and y_0 is the validation error of the FS_k model.

In AAT, the first step of the feature selection algorithm has a preset threshold τ_0 , set by AUTO-SET. So, we can get another training data point (x_1, y_1) in the pseudocode below) for our model after the first step of the feature selection algorithm. In forward feature selection, this data

point is the validation error of FS_1 and in backward selection FS_{k-2} .

We need one more training data point to make an initial prediction for k_{min}^* . We start updating the threshold one step later in the feature selection process to get another training data point. However, in backward selection this results in having 3 consecutive (in terms of their x-axis values) points. There can be many models that can fit these points. Some of these can be very far from the validation error distribution function we are trying to estimate. To avoid this, we use the average of the five best candidate feature sets FS_2^c for backward selection (or FS_{k-3}^c for forward selection) as an estimate data point. We use the average of the five best feature sets to avoid outliers in the first step of feature selection while the model is volatile.

ALGORITHM 2: Approximating validation error curve

```

1  $x_0, y_0 = k, validationError(FS_k)$ 
2  $FS_1 = runForwardSelectionStep(\tau)$ 
3  $x_1, y_1 = 1, validationError(FS_1)$ 
4  $x_2, y_2 = k - 1, AVG(validationError(best5(FS_2^c)))$ 
5  $X = x_0, x_1, x_2$ 
6  $Y = y_0, y_1, y_2$ 
7  $mlModel = polynomialRegression(degree = 2)$ 
8  $mlModel.fit(X, Y)$ 
9 for  $i \leftarrow 2$  to  $k$  do
10    $\tau = updateTh(mlModel)$ 
11    $FS_i = runFeatureSelectionStep(\tau)$ 
12    $x_{new}, y_{new} = i, validationError(FS_i)$ 
13    $X = X \cup \{x_{new}\}$ 
14    $Y = Y \cup \{y_{new}\}$ 
15    $mlModel.updateFit(X, Y)$ 
16 end

```

4.5.3 Forward Selection

This section describes how AAT utilizes the ML model for estimating the validation error curve during the feature selection process. This is the second step depicted in Figure 4.4. We first give the intuition of how we use the ML model and then describe in detail our algorithm.

Motivation. Depending on the setting of the feature selection procedure, there can be big variations on how the test error curve changes when the number of features changes. The setting of the procedure is the data, the ML model and the feature selection algorithm. For example, the number of truly informative features in the dataset determines the maximum possible point where the test error will stop decreasing. In addition, more complex relations between the informative features and how they interact with each other determine how much they contribute to reducing the test error and, as a result, the shape of the error curves. Another extreme example of a different setting is Decision Trees. Decision trees are models with high variance and low bias. Their model complexity is not linear to the number of feature; it is hard to quantify their complexity. Their error curves still are U-shaped, however the U-shape can be very different from other models. As a result of the high bias of the models, the minimum point can appear much faster. This makes it crucial to have a way to identify how the error curves change based on the number of features during the feature selection procedure in order to be able to improve it.

To solve this problem we use a learned model for the validation error curve. The test error is not known during the feature selection process, that is why we use the validation error as an approximation for the test error. Our algorithm tries to improve the performance of the feature selection process while also solving the problem of validation overfitting. When there is little validation overfitting, the validation error provides a good approximation of the test error.

A complex model has high variance. As a result, we want to use the validation dataset more there. We can achieve that by having a small threshold τ . Forward selection will have the its most complex model when it terminates, which is near the minimum point of the U-shaped validation error curve. Given this ML model, we can adjust the threshold of the feature selection algorithm.

Main Idea. Start with a threshold value τ_0 set by AUTO-SET and reduce it gradually as the model gets more complex. The minimum threshold value is τ_{min} at the predicted k_{min} point, which is the minimum point based on our ML learned validation curve.

Algorithm. Given the ML model, we want to find the k value where the minimum point is going to be. The model describes a second degree polynomial $a_2x^2 + a_1x + a_0$, so the predicted minimum point is going to be at $k_{min}^* = -a_1/2a_2$.

Next, we check if we will update the threshold τ at this step of the feature selection process. Because our ML model can be unstable, especially at first when it has very few training points, we add the constrain that $k_{min}^* \in [0.1k, 0.9k]$. Also, if we are at a step k_{cur} past k_{min} then we should stop decreasing the threshold because we have passed the point where we want the minimum τ_{min} .

τ_{min} is the minimum value of the threshold τ that we want during our feature selection process. For forward selection, τ_{min} is set to the initial value that AUTO-SET picks for backward selection. Then, we reduce gradually τ over the next $k_{min}^* - k_{cur}$ steps. As we update our ML model after each step, we have a new k_{min}^* after each step so the reduction of τ changes as well.

As we have previously mentioned in our analysis, we also adjust the noise parameter σ and we set it to $\tau/15$ at each step that we update τ .

The pseudocode for this update procedure for threshold τ can be see in Algorithm 3. This procedure is executed at every iteration of AAT; it is the second step in Figure 4.4.

ALGORITHM 3: updateTh() procedure for Forward Selection

Input : mlModel
1 $a_1, a_2 = mlModel.coefficient[0], mlModel.coefficient[1]$
2 $k_{min}^* = -a_1/2a_2$
3 **if** $k_{min}^* \in [0.1k, 0.9k]$ **and** $k_{cur} < k_{min}^*$ **then**
4 $\tau = \tau - \frac{\tau - \tau_{min}}{k_{min}^* - k_{cur}}$
5 $\sigma = \tau/15$
6 **end**

4.5.4 Backward Selection

For backward feature selection, the process of adjusting the threshold τ is very similar to before with a few key differences.

Backward selection starts with a complex model and simplifies it by removing features at each step. At the end of backward selection we have a simpler model than initially. A simple model has low variance. As a result, we do not need to use the validation set as much there as in the beginning. We achieve that by starting with the minimum threshold and gradually *increase* it. We want the largest threshold value τ_{max} at the minimum error point k_{min} .

Main Idea. Similarly to before, start with a threshold value τ_0 set by AUTO-SET and *increase* it gradually as the model gets simpler. The *maximum* threshold value is τ_{max} at the predicted k_{min}^* point, which is the point that we predict will have the minimum validation error.

Algorithm. Similarly to forward selection, we have the same minimum point k_{min}^* and we stop adjusting τ when we pass k_{min} .

However, the rule for updating the threshold τ is different. In this case, τ_{max} is the maximum value of τ that we want during the feature selection process. τ_{max} is set to the initial value that AUTO-SET picks for forward selection. The threshold increases at each step based on the following update rule:

$$\tau = \tau + \frac{\tau_{max} - \tau}{k_{cur} - k_{min}^*}$$

4.5.5 Wrapper methods

Wrapper methods for feature selection conduct a search in the space of all possible features. Forward and backward selection are two examples of wrapper methods. In general, wrapper methods require a state space, an initial space, a termination condition and a search engine [52].

Each state Q represents a set of features. The initial state Q_0 can be either the empty or the full feature set containing all k features. The search engine is a function that connect a state

Q_i to the next state Q_{i+1} by adding or deleting a feature. In our work we examine only stepwise wrapper methods that add or delete a only single feature at each step.

All such stepwise wrapper methods are a *combination* of steps of forward and backward selection. As a result, we can use AAT with all wrapper feature selection methods.

Main Idea. Start with a threshold value τ_0 set by AUTO-SET . At k_{min} we will have the minimum or maximum threshold value if the initial state Q_0 is the empty or the full feature set. At each step, *decrease* or *increase* the threshold if a feature gets added or deleted respectively.

ALGORITHM 4: updateTh() procedure for all Wrapper methods.

```

Input : mlModel
1  $a_1, a_2 = mlModel.coefficient[0], mlModel.coefficient[1]$   $k_{min}^* = -a_1/2a_2$ 
2 if  $k_{min}^* \in [0.1k, 0.9k]$  and  $k_{cur} < k_{min}^*$  then
3   if forward step then
4      $\tau = \tau - \frac{|\tau - \tau_0|}{|k_{min} - k_{cur}|}$ 
5   else
6      $\tau = \tau + \frac{|\tau - \tau_0|}{|k_{min} - k_{cur}|}$ 
7   end
8    $\sigma = \tau/15$ 
9 end

```

Algorithm. In Algorithm 4 we can see the generalization of the updateTh procedure for all wrapper methods. At each step the threshold is increased or decreased according to whether the model got more or less complex, i.e. whether a feature is added or deleted from the feature set. It is interesting to note that τ_0 has the the minimum or the maximum value of the threshold based on the initial state Q_0 .

4.6 Experiments

In this section we present our experiments. We begin by experimentally analyzing ThresholdOut algorithm [27]. We show that its performance depends on the user selected parameters.

Next, we show that AUTO-SET can pick these parameters automatically for the user and achieve close to the optimal solution. Lastly, we show that our feature selection algorithm Auto-Adjust Threshold can achieve an even lower test error in a lot of cases while reducing overfitting.

4.6.1 Experimental Design

We repeated our experiments for many different settings. To implement different machine learning models, we used python’s sklearn library. We used a naive Bayes model, a decision tree with the gini index and a minimum sample leaf of 10 and a logistic regression model with Newton’s method and a tolerance of 0.001.

For the feature selection algorithms, we implemented sequential forward selection (SFS) [51] and sequential backward selection (SBS) [51].

We used a synthetic dataset and multiple real datasets, as described below. We split each dataset into a training, a holdout/validation and a testing set consisting 50%, 25%, 25% of the whole dataset respectively.

For each different setting, consisting of a ML algorithm, a feature selection algorithm, a dataset and a different algorithm to avoid overfitting, we repeated the experiment 100 times and we report here average values for the results.

Synthetic Data. Initially, we created a synthetic dataset as a first step to verify experimentally our findings. We created $f = 60$ discrete features X_i with values 0 – 9. Of these f features, only $f_p = 20$ are predictive of the target variable Y , which is binary. All predictive features are conditionally independent and follow the distribution described by the probabilities: $p(X_i = 0|Y = 0) = p(X_i = 1|Y = 1) = p(X_i = 2|Y = 1) = \dots = p(X_i = 9|Y = 1) = 0.7$. All the other $f - f_p$ features are assigned values randomly. All features have a sample size of $n = 300$.

Real datasets. We used 4 different datasets from the UCI Machine Learning repository [6]. We used the heart dataset [4], the breast cancer dataset [3], the Ionosphere dataset [5] and the

Table 4.1. Characteristics of the datasets used.

Dataset	Number of features	Number of samples
Heart Disease dataset [4]	75	303
Z-Alizadeh Sani dataset [7]	56	303
Ionosphere dataset [5]	34	351
Breast Cancer Wisconsin [3]	32	569

Z-Alizadeh Sani dataset [7]. These datasets have 32-75 features and 300-550 examples. These datasets have the characteristics that motivated the problem of overfitting; a small number of samples and a large number of features for that number of samples. As we have mentioned these kind of datasets appear very often in the sciences, as it can be seen from the field of the four datasets we used. The characteristics of the datasets can be seen in Table 4.1.

Metrics. We capture a lot of different metrics to give a better picture of the performance of the algorithms throughout the feature selection process.

e_{res} is the test error of the process when the FS has converged.

o_{res} is the overfitting at the converged result as measured by the difference between the test and the validation error $e_t - e_v$.

o_{avg} is the average overfitting $e_t - e_v$ throughout the whole FS process, until we converge at the result

$I_e^{b_i}$ measures the percentage of steps of the FS process that the algorithm that we are comparing has a smaller test error than the baseline i we are comparing it against.

$D_e^{b_i}$ measures the average difference between the test errors of the algorithm we are comparing against the baseline i . A positive number means our algorithm is performing better on average.

$I_o^{b_i}$ is the percentage of steps of the FS process that our algorithm has less overfitting compared against the baseline i .

$D_o^{b_i}$ is the average difference between the overfitting of our algorithm and the baseline i over the whole FS process.

Table 4.2. Notations of metrics.

Metric	Meaning
e_{res}	Test error at converged result
o_{res}	Overfitting at converged result
o_{avg}	Average overfitting during FS
$I_e^{b_i}$	Percentage of times $e_t < e_t^{b_i}$
$D_e^{b_i}$	Average difference of $e_t^{b_i} - e_t$
$I_o^{b_i}$	Percentage of times $o < o^{b_i}$
$D_o^{b_i}$	Average difference of $o^{b_i} - o$

These metrics are summarized on Table 4.2.

4.6.2 Analysis of ThresholdOut

Experimental Setup. Our goal of this analysis is to examine how different values for the parameters of ThresholdOut τ and σ affect its performance during feature selection in different settings. For τ we examined the following values: [0.01, 0.05, 0.1, 0.15, 0.2]. For the noise σ we tested the values: [0.001, 0.01, 0.1]. As we see below these values give us a good picture of how they affect the results.

We performed our experiments for all the ML algorithms, feature selection algorithms and datasets described earlier. We repeated everything 100 times and we report the average values for $e_{res}, o_{res}, o_{avg}$ in each different setting. These results are in Tables 4.3 through Table 4.7.

Results. Tables 4.3 through Table 4.7 have all the results. In Figures 4.5, 4.6 and 4.7 we see some visualization of the test error (e_{res}) and the overfitting (o_{res}) for all the different values of τ and σ we tested in some different settings. Next, we point out our observations for all the different settings.

Both for forward and backward selection, a low value for τ and σ results in a lot of overfitting. This makes sense, as ThresholdOut with a value of $\tau = 0$ and $\sigma = 0$ is the same as

not using the algorithm. Small values for these parameters helped the problem of overfitting the least. Interestingly, depending on the setting, we achieved a small amount of overfitting for very different values. There was a big difference for forward and backward selection, but for different datasets as well.

Larger values for τ and a small σ have better performance for both forward and backward selection. However, for forward selection it is more obvious that a large value for τ gives the best performance. However, for backward selection it depends on the setting. For example, a value of $\tau = 0.05$ has the best performance on the ion and heart dataset for backward selection but on the synthetic dataset, a much larger value is required ($\tau = 0.15$). This is our motivation for AUTO-SET and AAT, to try and adapt based on the specific setting.

Large values for σ result in higher test error. Even though in this case there is very little overfitting. This makes sense as adding a lot of noise in the results that we return through ThresholdOut, results in making the wrong decisions a lot of the time and, as a result, getting a model with poor performance. However, because the feature selection is made based on a lot of randomness from the noise, there is little overfitting in the result. This can be seen in all the heatmap visualizations.

In Figure 4.2 we saw earlier a summary of our findings about the behavior of error and overfitting based on different values of τ and σ for ThresholdOut.

4.6.3 AUTO-SET

Experimental Setup. The goal of the following experiments is to show how AUTO-SET performs against manually picking the parameters for ThresholdOut τ and σ . We executed AUTO-SET in all the different settings that we executed the analysis of ThresholdOut in the previous section and we compare the results. The results can be seen in Table 4.3 through Table 4.7.

Results. All the results can be seen in Table 4.3 through Table 4.7. Some of the results of

AUTO-SET, along with the results of the previous analysis are visualized in Figure 4.8 through Figure 4.11. In these figures we see the test error and the overfitting for AUTO-SET and for ThresholdOut for different parameter values in different settings. For ThresholdOut, the size of the circle is proportionate to the value of τ and the opacity is proportionate to the value of σ .

In the visualized results, but also in all the results in the tables, for forward selection AUTO-SET is performing within 0.01 difference from the best test error of ThresholdOut. A lot of the times it has a smaller test error than the different values of τ and σ we picked.

In backward selection, in all cases the test error of AUTO-SET is within 0.006 difference of the best test error of ThresholdOut. However, we can see that in all backward selection cases the test error is pretty low compared to forward selection. That leaves little space for improvement. That is because backward selection starts with all features in the result set, including the most predictive ones. In our datasets, only few features have the most predictive power. So, backward selection cannot improve much by removing features with no predictive power and converges very soon.

4.6.4 Auto-Adjust Threshold

Experimental Setup. In this section we show how Auto-Adjust Threshold performs compared to the state-of-the-art algorithm, ThresholdOut, that tries to avoid overfitting a holdout dataset. Similarly to our previous experiments, we execute our algorithms multiple times in many different scenarios and report here the average results. Our results are summarized in Table 4.8.

Baselines. We compared our algorithm against two baselines. For the first baseline we executed all the feature selection algorithms without any process to protect against overfitting the holdout validation data. This baseline represents how most users implement their feature selection algorithms. For our second baseline, we used ThresholdOut to avoid overfitting the holdout dataset. For each different setting, we picked the values for τ and σ that performed best during our analysis of Thresholdout in Section 4.6.2.

Results. The full results of AAT can be seen in Table 4.8. In Figure 4.8 through Figure 4.11 we can also see the performance of AAT in comparison with AUTO-SET and ThresholdOut. We can see that AAT performs significantly better than both in most cases or performs equally well.

In Figures 4.12, 4.13, 4.14 and 4.15 we see I_e and I_o for both baselines. For baseline 1, in almost all cases there's an at least 80% improvement in overfitting in all steps and settings. This makes sense as baseline 1 does not control for overfitting. As a result, we can see that I_e is improving in all cases in at least 50% of feature selection steps. Baseline 2 has already very little overfitting because it uses ThresholdOut, so as expected we there isn't much of an improvement in overfitting as seen in Figure 4.15. However, because AAT adjusts the threshold throughout the feature selection process, we see that I_e is improving in all cases in at least 50% of feature selection steps.

The improvement we see in the error depends on the dataset. On the heart dataset, the test error on Naive Bayes with ThresholdOut is 0.345 and with AAT it is 0.311, which is a 10% improvement. And AAT has 60% less overfitting. Similarly, with logistic regression the error with ThresholdOut is 0.315 and with AAT it is 0.293, which is a 7% improvement. AAT has 50% less overfitting in the result.

4.7 Related Work

The problem of overfitting the validation dataset has been identified before [65, 67]. The authors showed that the more complex the wrapper feature selection algorithm is, the more the potential for overfitting. They showed that even with cross-validation and only 50 features and 300 samples, there can be significant overfitting even in a simple feature selection algorithm, such as sequential feature selection. This leads to decreased performance. It is a well know fact

from statistics that the more hypotheses are tested against the same data, the higher the possibility of having false discoveries. The equivalent statement for ML was shown in [56]. The authors showed that the more feature sets an algorithm tests, the higher the possibility of overfitting the holdout dataset. The problem of overfitting is not limited to feature selection, but it can occur in any step of model selection, such as hyperparameter tuning [21].

Suggested solutions to the problem of overfitting the validation dataset are similar to the solutions to the problem of avoiding overfitting the training dataset. Namely, regularisation [20], early stopping [55, 64], model and hyperparameter averaging [19, 36]. In addition, it has been recommended to avoid complex model selection processes when the datasets are small [21]. In these cases Bayesian approaches [84] or ensemble approaches, like random forests [17], can be used because they avoid model selection.

A novel approach to solving this problem was proposed in the ThresholdOut paper [27, 28, 29]. We have discussed this paper extensively in Section 4.2.2. This work has been strengthened qualitatively [9] and quantitatively [62]. These works make the algorithm applicable to a broader range of queries and give tighter bounds for its guarantees. Regarding all these approaches, he showed in Section 4.3 the main shortcoming of ThresholdOut that make it not practical. In addition, as we have already talked about, our algorithm AAT is designed specifically for feature selection and can take advantage of the structure of the queries to achieve better results.

Another work based on [27], showed how to reuse a holdout dataset specifically for accurate leaderboards in machine learning competitions, where users submit adaptively chosen models in the process of a competition [14]. This work is similar to our in the way that it is inspired by the ThresholdOut algorithm, and it improved it for a very specific case.

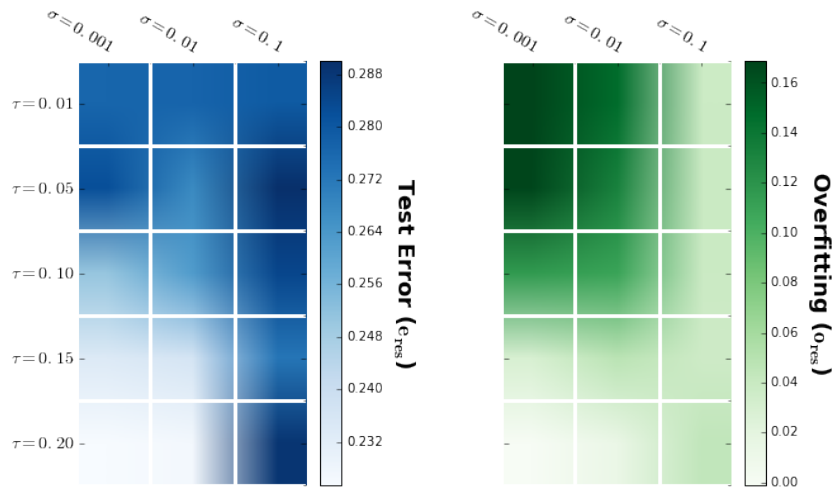
4.8 Conclusion

In this chapter, we presented our work on the problem of overfitting during feature selection. We experimentally analyzed the state of the art approach to solve the problem and showed that its performance is dependent on its input parameters. Then, we presented AUTO-SET, an automated way to set this parameters that achieves near optimal performance. Lastly, based on our lessons we presented Auto Adjust Threshold, an algorithm for feature selection that avoids overfitting the holdout dataset. Our experimental analysis showed that AAT can reduce significantly both overfitting and the error.

Acknowledgements

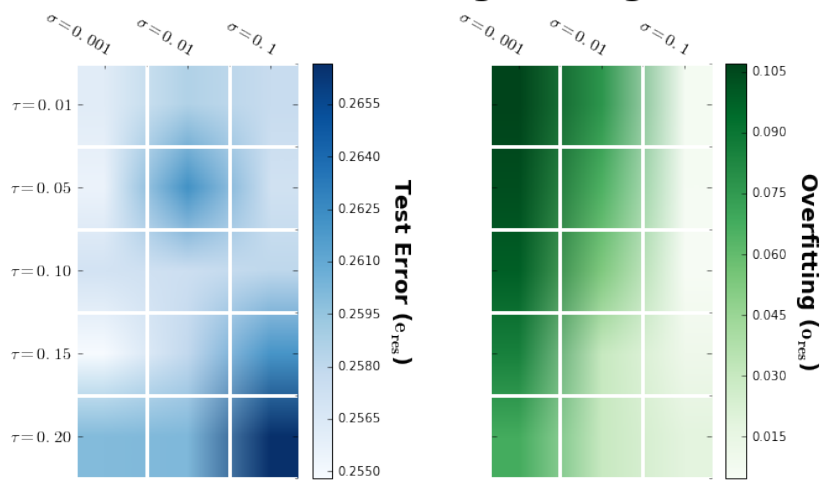
This chapter contains material adapted from the paper “A practical feature selection algorithm to avoid validation overfitting in autoML systems” by Nikolaos (Nikos) Koulouris; Arun Kumar; Yannis Papakonstantinou. It is currently being prepared for publication. The dissertation author was the primary investigator and author of this paper.

Forward Selection with Logistic Regression



(a) Forward Selection

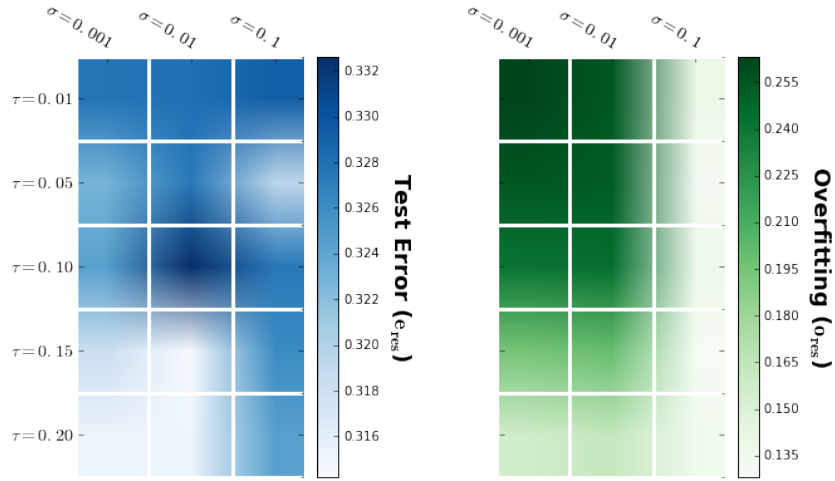
Backward Selection with Logistic Regression



(b) Backward Selection

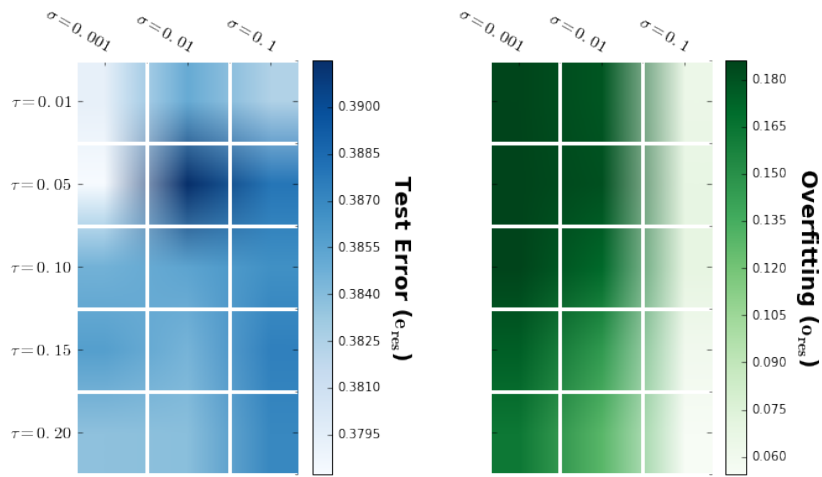
Figure 4.5. Heatmap of test error (e_{res}) and overfitting (o_{res}) for forward and backward selection in the synthetic dataset.

Forward Selection with Logistic Regression



(a) Forward Selection

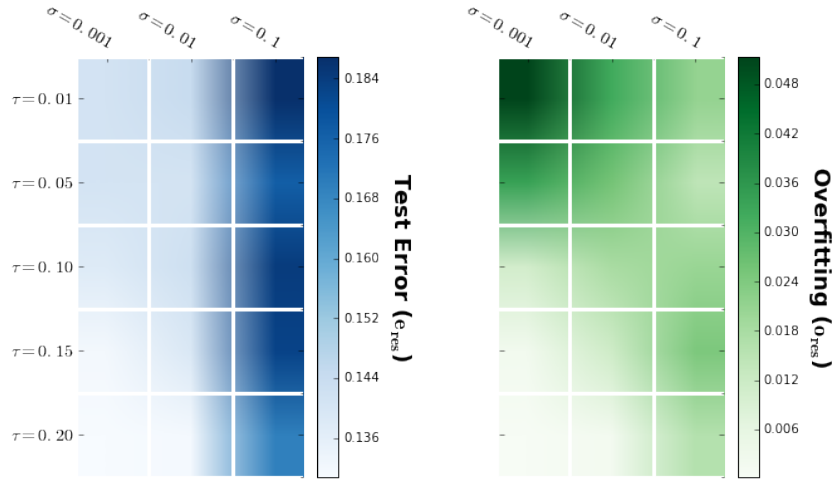
Backward Selection with Logistic Regression



(b) Backward Selection

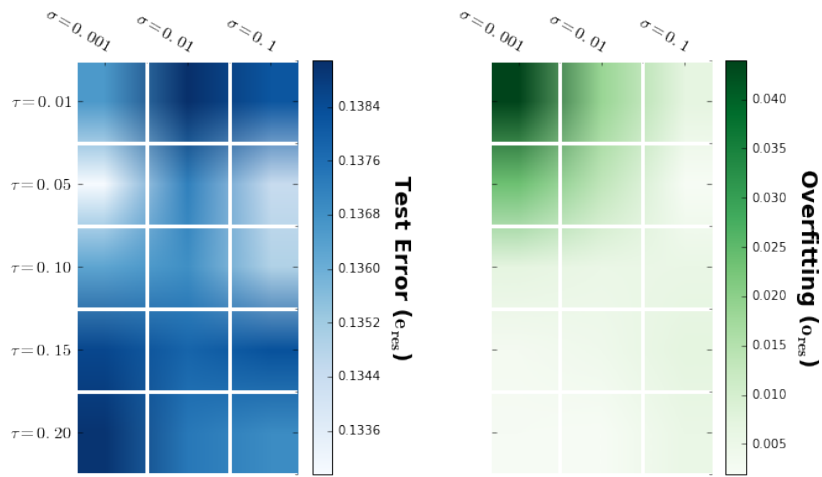
Figure 4.6. Heatmap of test error (e_{res}) and overfitting (o_{res}) for forward and backward selection in the heart dataset.

Forward Selection with Logistic Regression



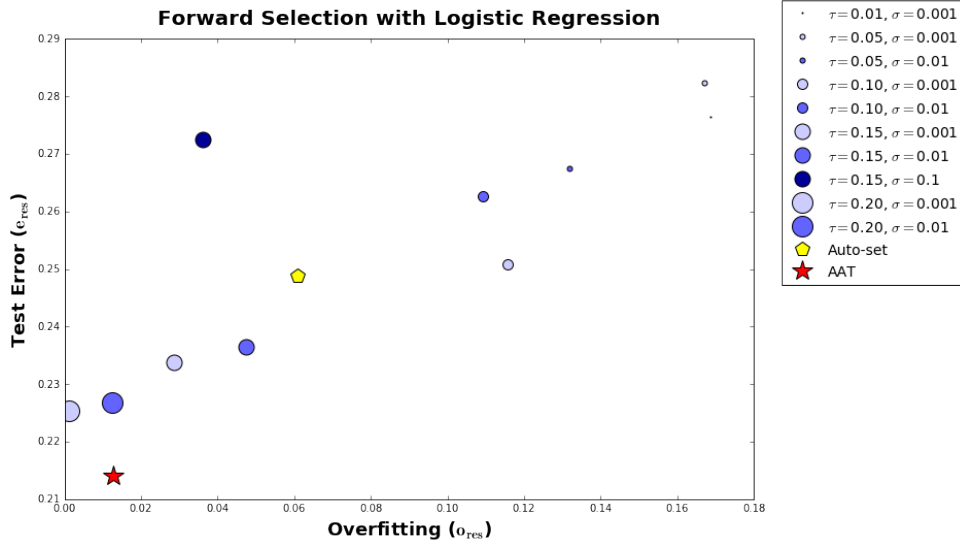
(a) Forward Selection

Backward Selection with Logistic Regression

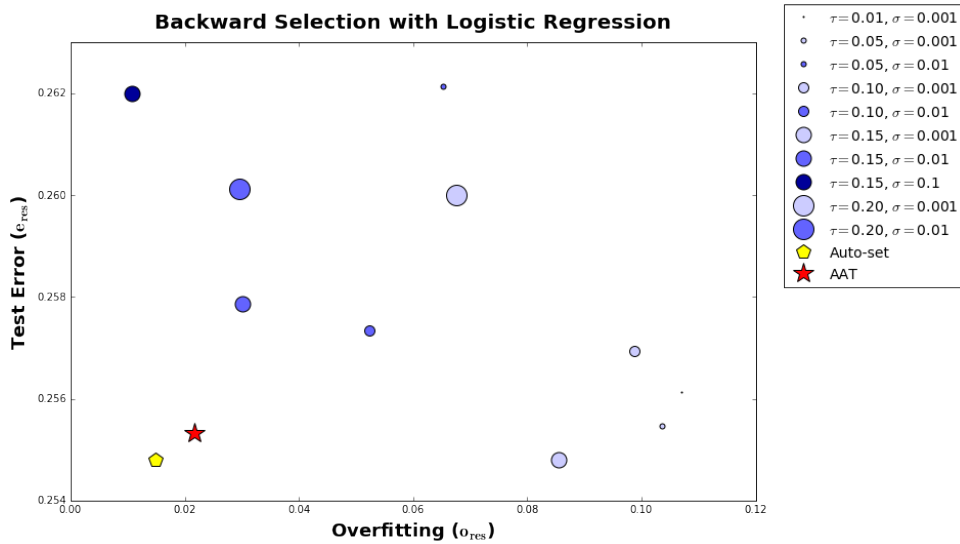


(b) Backward Selection

Figure 4.7. Heatmap of test error (e_{res}) and overfitting (o_{res}) for forward and backward selection in the ion dataset.

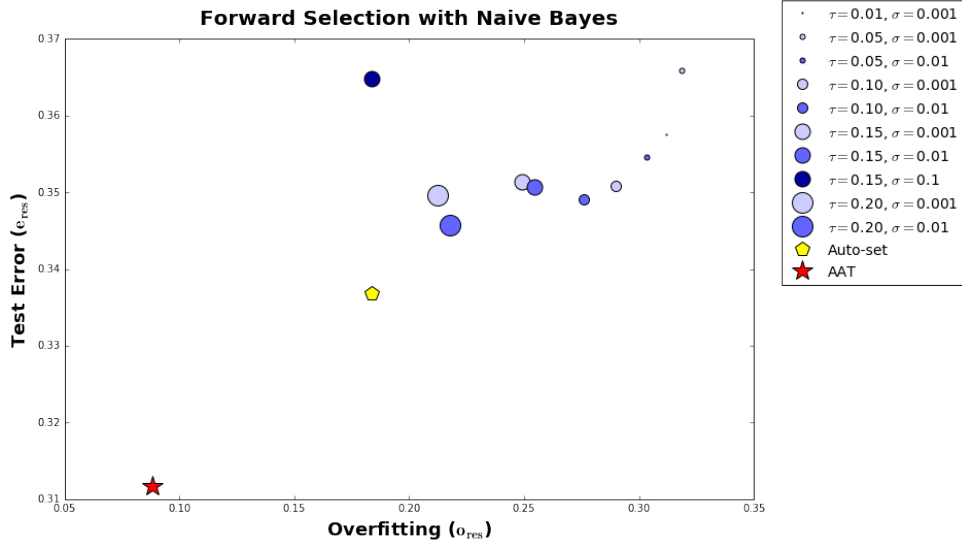


(a) Forward Selection

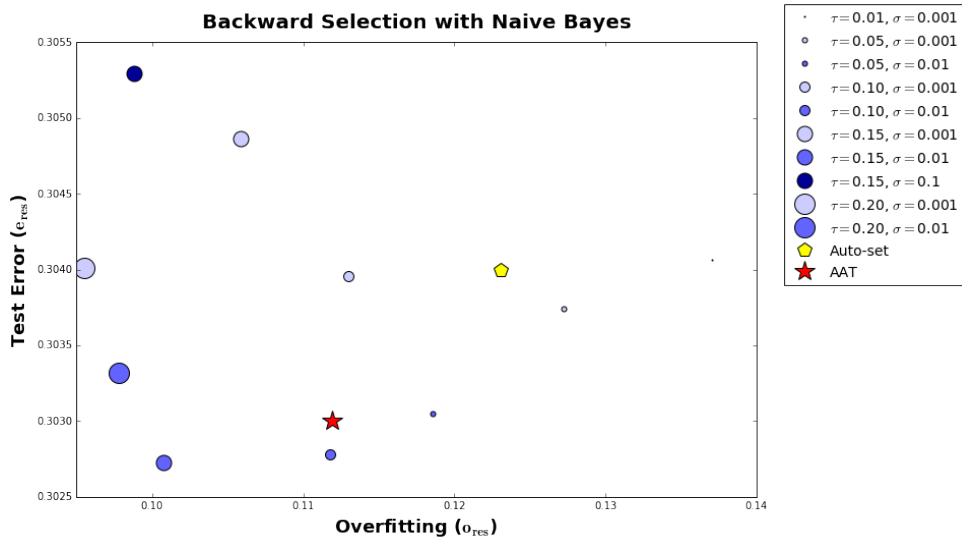


(b) Backward Selection

Figure 4.8. Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the synthetic dataset with Logistic Regression.

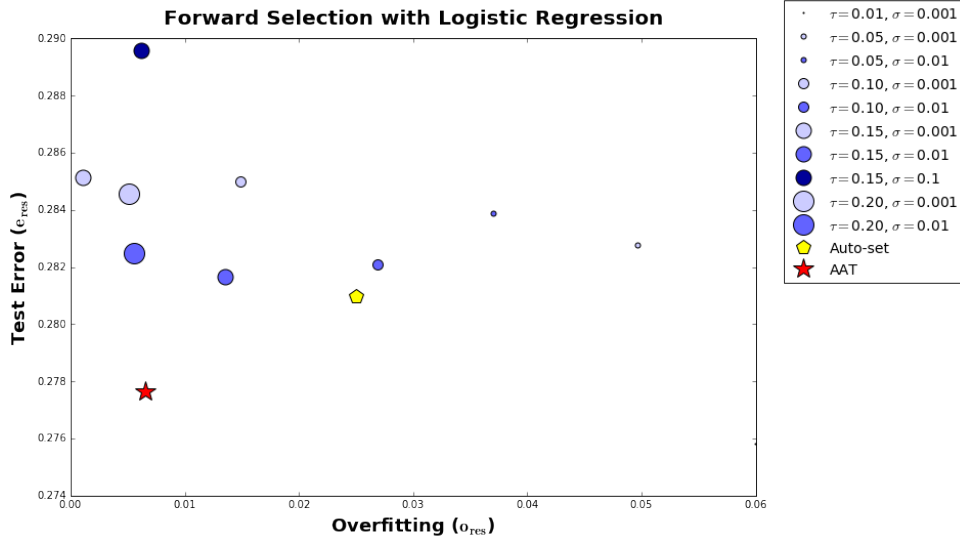


(a) Forward Selection

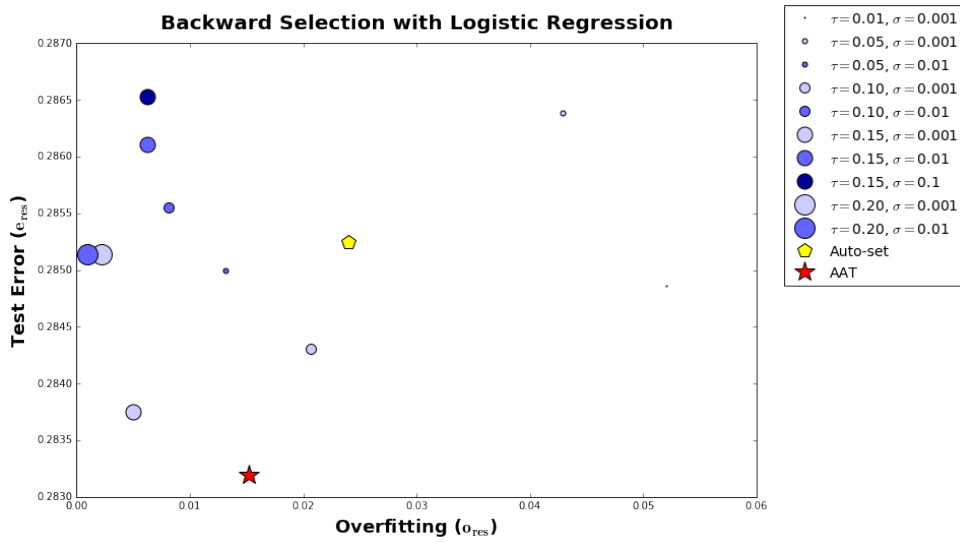


(b) Backward Selection

Figure 4.9. Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the heart dataset with Naive Bayes.

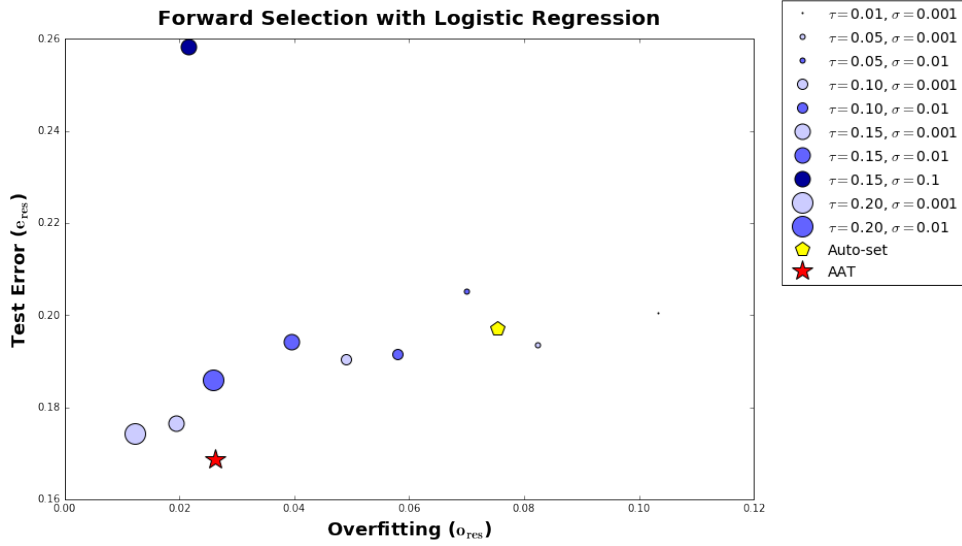


(a) Forward Selection

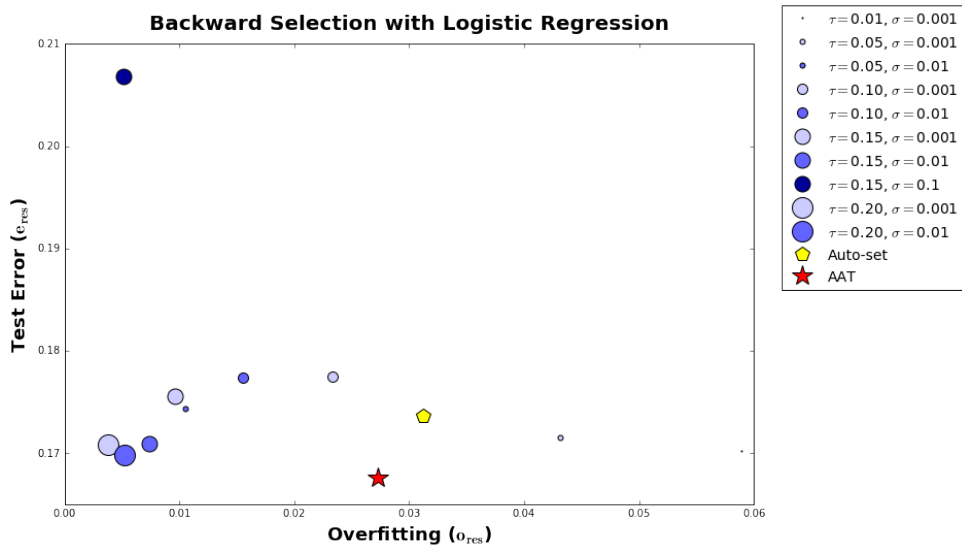


(b) Backward Selection

Figure 4.10. Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the cancer dataset with Logistic Regression.



(a) Forward Selection



(b) Backward Selection

Figure 4.11. Test error (e_{res}) and overfitting (o_{res}) at the result of Forward Selection for different parameters of ThresholdOut and our approaches on the sani dataset with Logistic Regression.

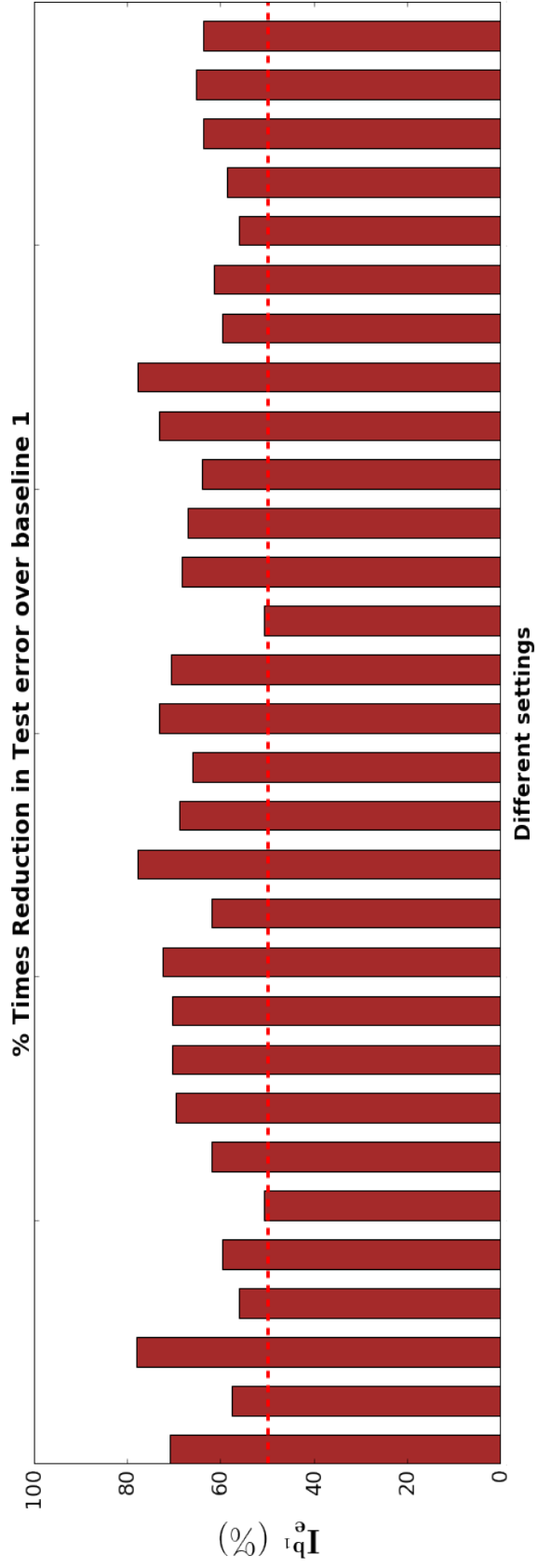


Figure 4.12. Percentage of times $e_t < e_t^{b_1}$ in all 30 different settings.

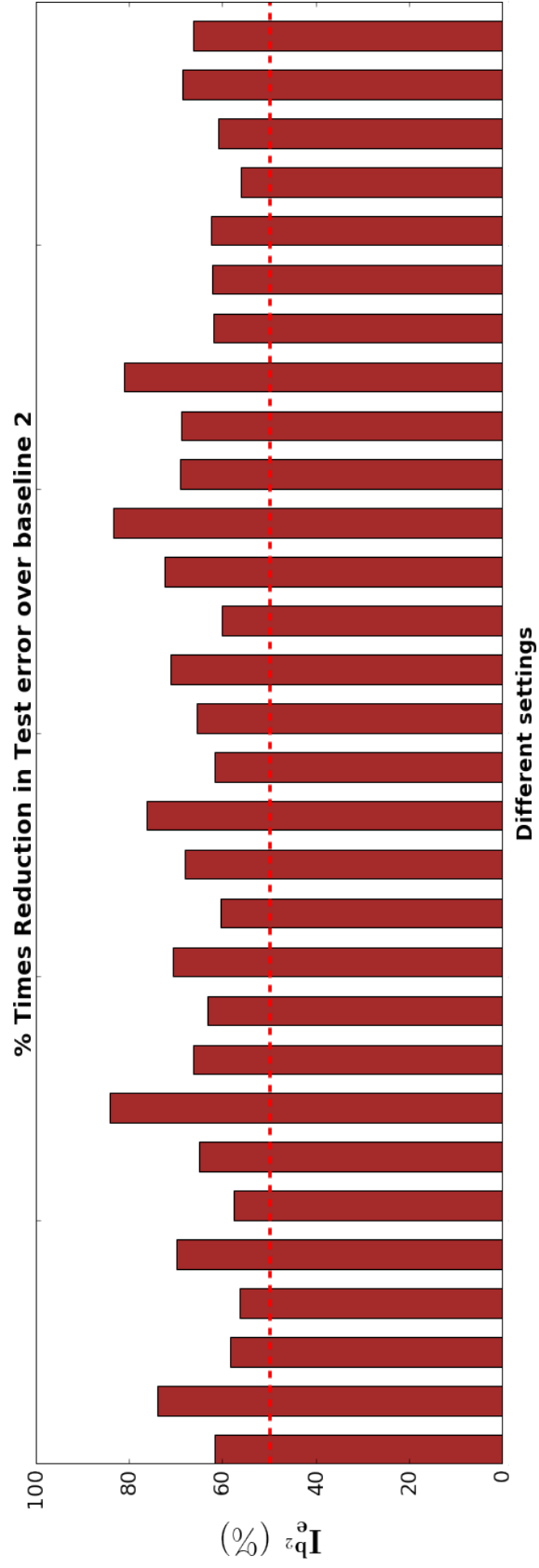


Figure 4.14. Percentage of times $e_t < e_t^{b_2}$ in all 30 different settings.

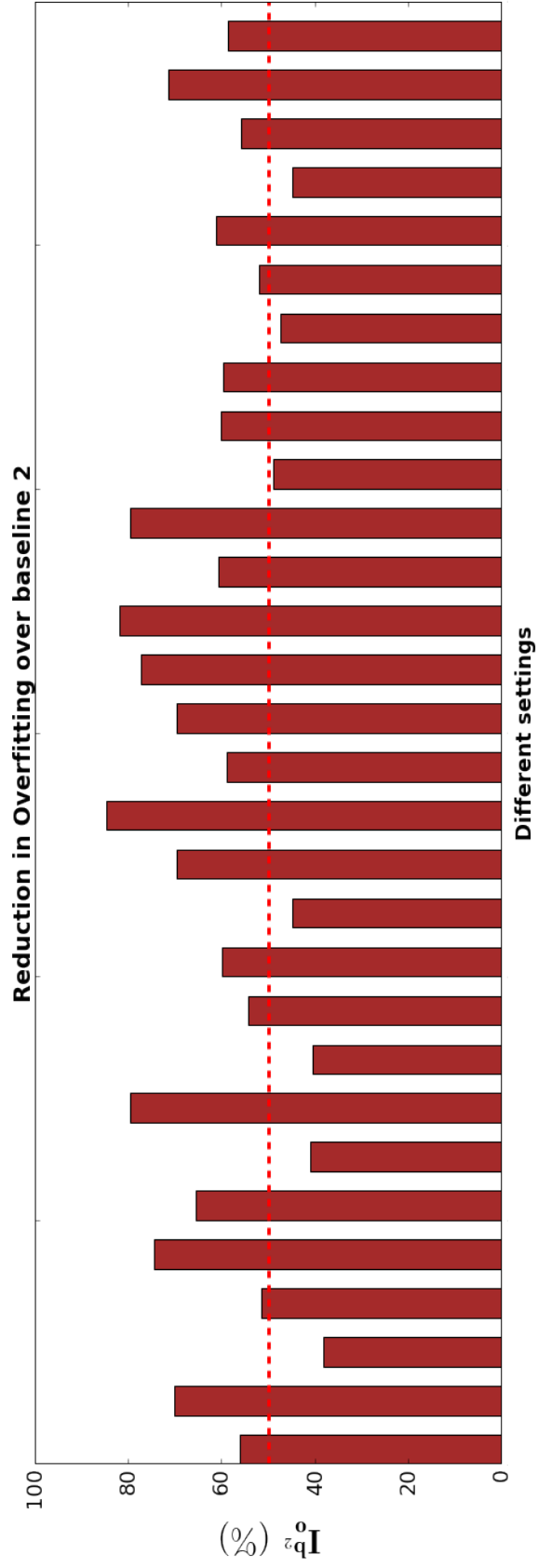


Figure 4.15. Percentage of times $o < o^{b2}$ in all 30 different settings.

Table 4.3. Results of ThresholdOut analysis on synthetic data.

ML Algorithm	τ	σ	Forward Selection			Backward Selection		
			ϵ_{res}	θ_{res}	θ_{avg}	ϵ_{res}	θ_{res}	θ_{avg}
Naive Bayes	0.01	0.01	0.287	0.145	0.110	0.227	0.024	0.066
Naive Bayes	0.01	0.1	0.288	0.040	0.020	0.227	0.001	0.014
Naive Bayes	0.1	0.01	0.280	0.102	0.070	0.229	0.012	0.025
Naive Bayes	0.1	0.1	0.305	0.041	0.021	0.226	0.001	0.013
Naive Bayes	0.2	0.01	0.276	0.031	0.018	0.223	0.005	0.001
Naive Bayes	0.2	0.1	0.309	0.050	0.021	0.227	0.001	0.008
Naive Bayes	AUTO-SET		0.286	0.060	0.039	0.228	0.034	0.068
Decision Trees	0.01	0.01	0.249	0.108	0.084	0.375	0.047	0.078
Decision Trees	0.01	0.1	0.279	0.037	0.026	0.383	0.009	0.016
Decision Trees	0.1	0.01	0.245	0.065	0.035	0.374	0.023	0.049
Decision Trees	0.1	0.1	0.288	0.045	0.014	0.378	0.009	0.019
Decision Trees	0.2	0.01	0.227	0.003	0.001	0.375	0.016	0.019
Decision Trees	0.2	0.1	0.279	0.034	0.012	0.382	0.014	0.022
Decision Trees	AUTO-SET		0.232	0.009	0.006	0.374	0.026	0.040
Logistic Regression	0.01	0.01	0.277	0.147	0.099	0.258	0.077	0.082
Logistic Regression	0.01	0.1	0.279	0.037	0.019	0.257	0.006	0.015
Logistic Regression	0.1	0.01	0.262	0.109	0.065	0.257	0.052	0.049
Logistic Regression	0.1	0.1	0.284	0.038	0.019	0.258	0.005	0.012
Logistic Regression	0.2	0.01	0.226	0.012	0.005	0.260	0.029	0.027
Logistic Regression	0.2	0.1	0.288	0.044	0.020	0.266	0.017	0.016
Logistic Regression	AUTO-SET		0.248	0.060	0.033	0.254	0.015	0.019

Table 4.4. Results of ThresholdOut analysis on sani dataset.

ML Algorithm	τ	σ	Forward Selection			Backward Selection		
			ϵ_{res}	$\mathbf{0}_{res}$	$\mathbf{0}_{avg}$	ϵ_{res}	$\mathbf{0}_{res}$	$\mathbf{0}_{avg}$
Naive Bayes	0.01	0.01	0.211	0.078	0.061	0.251	0.041	0.046
Naive Bayes	0.01	0.1	0.258	0.022	0.016	0.273	0.016	0.021
Naive Bayes	0.1	0.01	0.201	0.044	0.034	0.240	0.024	0.023
Naive Bayes	0.1	0.1	0.273	0.021	0.016	0.277	0.021	0.017
Naive Bayes	0.2	0.01	0.192	0.017	0.008	0.234	0.013	0.009
Naive Bayes	0.2	0.1	0.275	0.032	0.017	0.270	0.012	0.019
Naive Bayes	AUTO-SET		0.205	0.059	0.042	0.241	0.063	0.068
Decision Trees	0.01	0.01	0.212	0.056	0.036	0.188	0.012	0.022
Decision Trees	0.01	0.1	0.275	0.020	0.012	0.190	0.001	0.009
Decision Trees	0.1	0.01	0.195	0.0219	0.013	0.193	0.005	0.007
Decision Trees	0.1	0.1	0.273	0.014	0.010	0.192	0.002	0.002
Decision Trees	0.2	0.01	0.189	0.001	0.003	0.197	0.001	0.001
Decision Trees	0.2	0.1	0.276	0.023	0.008	0.192	0.001	0.006
Decision Trees	AUTO-SET		0.190	0.016	0.008	0.182	0.016	0.021
Logistic Regression	0.01	0.01	0.204	0.075	0.058	0.178	0.016	0.024
Logistic Regression	0.01	0.1	0.261	0.023	0.014	0.206	0.001	0.009
Logistic Regression	0.1	0.01	0.191	0.058	0.040	0.177	0.015	0.016
Logistic Regression	0.1	0.1	0.264	0.036	0.016	0.207	0.002	0.013
Logistic Regression	0.2	0.01	0.185	0.025	0.010	0.169	0.005	0.004
Logistic Regression	0.2	0.1	0.265	0.034	0.018	0.194	0.001	0.006
Logistic Regression	AUTO-SET		0.197	0.075	0.059	0.173	0.031	0.038

Table 4.5. Results of ThresholdOut analysis on heart dataset.

ML Algorithm	τ	σ	Forward Selection			Backward Selection		
			e_{res}	o_{res}	o_{avg}	e_{res}	o_{res}	o_{avg}
Naive Bayes	0.01	0.01	0.354	0.305	0.239	0.304	0.127	0.179
Naive Bayes	0.01	0.1	0.366	0.192	0.143	0.305	0.105	0.125
Naive Bayes	0.1	0.01	0.349	0.276	0.214	0.303	0.112	0.145
Naive Bayes	0.1	0.1	0.370	0.215	0.153	0.303	0.097	0.107
Naive Bayes	0.2	0.01	0.345	0.217	0.170	0.303	0.097	0.107
Naive Bayes	0.2	0.1	0.370	0.194	0.148	0.304	0.101	0.118
Naive Bayes	AUTO-SET		0.336	0.183	0.143	0.306	0.123	0.165
Decision Trees	0.01	0.01	0.344	0.209	0.150	0.288	0.031	0.063
Decision Trees	0.01	0.1	0.353	0.115	0.058	0.289	0.038	0.017
Decision Trees	0.1	0.01	0.330	0.161	0.098	0.285	0.005	0.042
Decision Trees	0.1	0.1	0.344	0.109	0.043	0.290	0.025	0.022
Decision Trees	0.2	0.01	0.312	0.066	0.012	0.301	0.001	0.008
Decision Trees	0.2	0.1	0.340	0.101	0.024	0.290	0.025	0.011
Decision Trees	AUTO-SET		0.300	0.011	0.005	0.292	0.040	0.042
Logistic Regression	0.01	0.01	0.328	0.254	0.223	0.385	0.178	0.196
Logistic Regression	0.01	0.1	0.329	0.138	0.088	0.382	0.064	0.079
Logistic Regression	0.1	0.01	0.332	0.244	0.197	0.385	0.171	0.162
Logistic Regression	0.1	0.1	0.327	0.136	0.088	0.386	0.069	0.070
Logistic Regression	0.2	0.01	0.315	0.162	0.135	0.383	0.128	0.118
Logistic Regression	0.2	0.1	0.324	0.132	0.081	0.387	0.054	0.058
Logistic Regression	AUTO-SET		0.300	0.109	0.0825	0.383	0.077	0.084

Table 4.6. Results of ThresholdOut analysis on ion dataset.

ML Algorithm	τ	σ	Forward Selection			Backward Selection		
			e_{res}	o_{res}	o_{avg}	e_{res}	o_{res}	o_{avg}
Naive Bayes	0.01	0.01	0.115	0.039	0.028	0.116	0.014	0.020
Naive Bayes	0.01	0.1	0.168	0.016	0.013	0.130	0.011	0.012
Naive Bayes	0.1	0.01	0.114	0.024	0.014	0.118	0.012	0.015
Naive Bayes	0.1	0.1	0.167	0.025	0.015	0.120	0.008	0.009
Naive Bayes	0.2	0.01	0.105	0.007	0.004	0.121	0.010	0.006
Naive Bayes	0.2	0.1	0.163	0.018	0.011	0.124	0.005	0.009
Naive Bayes	AUTO-SET		0.106	0.016	0.011	0.118	0.026	0.031
Decision Trees	0.01	0.01	0.142	0.0377	0.023	0.118	0.009	0.016
Decision Trees	0.01	0.1	0.175	0.026	0.012	0.121	0.005	0.009
Decision Trees	0.1	0.01	0.133	0.016	0.007	0.126	0.004	0.003
Decision Trees	0.1	0.1	0.156	0.021	0.008	0.119	0.003	0.004
Decision Trees	0.2	0.01	0.132	0.005	0.002	0.125	0.003	0.005
Decision Trees	0.2	0.1	0.151	0.144	0.103	0.121	0.002	0.008
Decision Trees	AUTO-SET		0.134	0.013	0.007	0.111	0.011	0.015
Logistic Regression	0.01	0.01	0.144	0.032	0.030	0.139	0.018	0.024
Logistic Regression	0.01	0.1	0.186	0.021	0.011	0.138	0.006	0.010
Logistic Regression	0.1	0.01	0.142	0.018	0.014	0.136	0.005	0.009
Logistic Regression	0.1	0.1	0.184	0.020	0.010	0.134	0.005	0.008
Logistic Regression	0.2	0.01	0.131	0.002	0.002	0.137	0.002	0.001
Logistic Regression	0.2	0.1	0.169	0.016	0.009	0.136	0.005	0.008
Logistic Regression	AUTO-SET		0.146	0.025	0.019	0.136	0.0313	0.031

Table 4.7. Results of ThresholdOut analysis on cancer dataset.

ML Algorithm	τ	σ	Forward Selection			Backward Selection		
			ϵ_{res}	O_{res}	O_{avg}	ϵ_{res}	O_{res}	O_{avg}
Naive Bayes	0.01	0.01	0.308	0.087	0.069	0.421	0.028	0.032
Naive Bayes	0.01	0.1	0.312	0.027	0.021	0.481	0.001	0.004
Naive Bayes	0.1	0.01	0.313	0.051	0.043	0.405	0.001	0.005
Naive Bayes	0.1	0.1	0.319	0.030	0.026	0.472	0.014	0.010
Naive Bayes	0.2	0.01	0.319	0.015	0.013	0.393	0.003	0.002
Naive Bayes	0.2	0.1	0.309	0.023	0.022	0.458	0.009	0.012
Naive Bayes	AUTO-SET		0.310	0.052	0.045	0.394	0.029	0.034
Decision Trees	0.01	0.01	0.281	0.037	0.027	0.281	0.024	0.029
Decision Trees	0.01	0.1	0.289	0.011	0.008	0.278	0.012	0.014
Decision Trees	0.1	0.01	0.276	0.014	0.009	0.282	0.014	0.017
Decision Trees	0.1	0.1	0.287	0.012	0.009	0.280	0.011	0.014
Decision Trees	0.2	0.01	0.273	0.001	0.004	0.279	0.007	0.010
Decision Trees	0.2	0.1	0.294	0.014	0.004	0.278	0.009	0.011
Decision Trees	AUTO-SET		0.272	0.027	0.023	0.277	0.026	0.030
Logistic Regression	0.01	0.01	0.279	0.039	0.034	0.285	0.015	0.030
Logistic Regression	0.01	0.1	0.289	0.005	0.001	0.286	0.006	0.010
Logistic Regression	0.1	0.01	0.282	0.027	0.018	0.286	0.008	0.015
Logistic Regression	0.1	0.1	0.282	0.002	0.002	0.287	0.004	0.013
Logistic Regression	0.2	0.01	0.283	0.006	0.002	0.285	0.001	0.004
Logistic Regression	0.2	0.1	0.289	0.004	0.001	0.287	0.005	0.009
Logistic Regression	AUTO-SET		0.286	0.045	0.039	0.286	0.024	0.031

Table 4.8. Performance of AAT in different settings.

Dataset	Feature Selection	ML Algo	e_{res}	\mathbf{o}_{res}	I_c^{b1}	D_c^{b1}	I_0^{b1}	D_0^{b1}	I_c^{b2}	D_c^{b2}	I_0^{b2}	D_0^{b2}
Synthetic	Forward	Naive Bayes	0.259	0.020	70.9	0.029	93.0	0.106	61.6	0.010	5.59	0.012
Synthetic	Forward	Decision Trees	0.227	0.008	57.6	0.010	86.8	0.09	74.0	0.003	70.0	0.001
Synthetic	Forward	Logistic Regression	0.214	0.012	78.1	0.049	87.1	0.086	58.4	0.005	38.2	-0.020
Synthetic	Backward	Naive Bayes	0.232	0.032	55.9	0.001	91.6	0.086	56.2	0.002	51.5	-0.001
Synthetic	Backward	Decision Trees	0.375	0.029	59.5	0.001	82.8	0.058	69.8	0.003	74.5	0.020
Synthetic	Backward	Logistic Regression	0.255	0.021	50.7	0.007	91.8	0.091	57.5	0.002	65.6	0.023
Sani	Forward	Naive Bayes	0.186	0.008	61.9	0.007	83.5	0.048	65.0	0.007	40.8	0.018
Sani	Forward	Decision Trees	0.183	0.001	69.6	0.017	87.4	0.056	84.1	0.007	79.6	0.001
Sani	Forward	Logistic Regression	0.168	0.026	70.3	0.023	83.6	0.049	66.2	0.014	40.4	-0.014
Sani	Backward	Naive Bayes	0.235	0.038	70.4	0.020	87.3	0.053	63.3	0.009	54.2	0.002
Sani	Backward	Decision Trees	0.188	0.019	72.3	0.008	81.1	0.004	70.6	0.003	59.8	0.011
Sani	Backward	Logistic Regression	0.167	0.027	61.8	0.004	81.0	0.039	60.4	0.004	44.8	-0.012
Heart	Forward	Naive Bayes	0.311	0.088	77.7	0.032	91.9	0.118	68.1	0.0193	69.5	0.065
Heart	Forward	Decision Trees	0.301	0.016	68.7	0.043	90.3	0.185	76.2	0.017	84.7	0.077
Heart	Forward	Logistic Regression	0.293	0.058	66.0	0.024	79.7	0.097	61.7	0.015	58.9	0.031
Heart	Backward	Naive Bayes	0.304	0.112	73.2	0.015	86.3	0.067	65.5	0.010	69.7	0.037
Heart	Backward	Decision Trees	0.286	0.027	70.7	0.001	82.6	0.042	71.1	0.001	77.3	0.021
Heart	Backward	Logistic Regression	0.386	0.085	50.7	0.004	91.2	0.115	60.1	0.008	81.9	0.086
Ion	Forward	Naive Bayes	0.103	0.008	68.4	0.007	86.2	0.029	72.5	0.004	60.5	0.002
Ion	Forward	Decision Trees	0.130	0.005	67.0	0.006	82.6	0.033	83.5	0.001	79.6	0.004
Ion	Forward	Logistic Regression	0.133	0.004	63.9	0.004	85.5	0.033	69.2	0.003	48.8	0.012
Ion	Backward	Naive Bayes	0.118	0.013	73.2	0.009	86.8	0.026	68.7	0.005	60.0	0.002
Ion	Backward	Decision Trees	0.114	0.007	77.8	0.003	83.7	0.014	81.0	0.012	59.6	0.008
Ion	Backward	Logistic Regression	0.136	0.022	59.7	0.001	84.5	0.032	62.0	0.002	47.3	0.009
Cancer	Forward	Naive Bayes	0.314	0.019	58.5	0.005	86.8	0.061	66.3	0.003	50.2	0.003
Cancer	Forward	Decision Trees	0.274	0.008	65.3	0.005	84.2	0.035	73.6	0.004	57.4	0.011
Cancer	Forward	Logistic Regression	0.277	0.007	69.8	0.010	80.9	0.042	65.8	0.003	35.9	-0.013
Cancer	Backward	Naive Bayes	0.417	0.027	63.4	0.030	79.1	0.029	59.8	0.007	51.3	0.010
Cancer	Backward	Decision Trees	0.278	0.015	73.3	0.006	85.9	0.015	73.0	0.001	62.3	0.001
Cancer	Backward	Logistic Regression	0.283	0.015	70.5	0.006	86.6	0.033	64.4	0.003	44.4	-0.008

Chapter 5

Conclusion

Data deluge has created the opportunity to transition to data-driven research in many fields. This means more opportunities to test more hypotheses. However, this opportunity also has the caveat of false discoveries because of the problem of multiple comparisons. In data exploration, the more hypotheses we test the higher the chance of making a statement purely by chance. In machine learning, the more models that we test during model selection the higher the chance that we will have an overfitted model.

In this dissertation examined the multiple comparisons problem in data exploration and machine learning. In both cases, we proposed novel techniques that exploit the structure that each problem has in order to control false discoveries and achieve improved performance.

Hierarchical structure is very common in many different datasets. In Chapter 3 we presented *VigilaDE*, a novel way to perform data exploration that exploits the hierarchical structure of the input data. *VigilaDE* has two novel data exploration algorithms that use the hierarchical structure to guide the data exploration in a top-down approach, exploring only part of the input data. This results in avoiding testing all possible hypotheses. Testing less hypotheses means increased statistical power. We showed with synthetic and real-world datasets that our approach can lead in an increase of up to 2.7x in statistical power.

In many different machine learning scenarios models are chosen adaptively until the best one is found. Some examples are feature selection, hyperparameter tuning and boosting.

In such scenarios existing techniques from statistics cannot be used to avoid overfitting. In Chapter 4 we analyzed ThresholdOut, the state-of-the-art solution for this problem and proposed a novel technique for feature selection. We showed that a big shortcoming of ThresholdOut is its requirement for the users to pick its parameters. These parameters can influence the effectiveness of ThresholdOut a lot. We proposed AUTO-SET, an automated way to set these parameters for feature selection. AUTO-SET tries to pick the best parameters for every different setting. We showed that in most cases it performs as well or better than most manually picked parameter values. The problem of feature selection has some structure. The models evaluated are increasingly more or less complex. We proposed Auto Adjust Threshold (AAT), a novel way to perform feature selection. AAT utilizes this structure as captured by the bias-variance trade-off to automatically adjust a threshold throughout the feature selection process. We showed that AAT can achieve a significant reduction in generalization error while reducing overfitting.

Bibliography

- [1] Amazon sagemaker autopilot. <https://aws.amazon.com/sagemaker/autopilot/>. Accessed: 2020-02-20.
- [2] Google automl. <https://cloud.google.com/automl>. Accessed: 2020-02-20.
- [3] Uci breast cancer wisconsin (diagnostic) dataset. [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)). Accessed: 2020-02-20.
- [4] Uci heart disease dataset. <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>. Accessed: 2020-02-20.
- [5] Uci ionosphere dataset. <https://archive.ics.uci.edu/ml/datasets/Ionosphere>. Accessed: 2020-02-20.
- [6] Uci machine learning repository. <https://archive.ics.uci.edu/ml/datasets.php>. Accessed: 2020-02-20.
- [7] Uci z-alizadeh sani dataset. <https://archive.ics.uci.edu/ml/datasets/Z-Alizadeh+Sani>. Accessed: 2020-02-20.
- [8] Rina Foygel Barber and Emmanuel J Candès. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 2015.
- [9] Raef Bassily, Adam Smith, Thomas Steinke, and Jonathan Ullman. More general queries and less generalization error in adaptive data analysis. *arXiv preprint arXiv:1503.04843*, 2015.
- [10] Sebastian Bauer, Steffen Grossmann, Martin Vingron, and Peter N Robinson. Ontologizer 2.0 - a multifunctional tool for go term enrichment analysis and data exploration. *Bioinformatics*, 2008.
- [11] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society*, 1995.
- [12] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, 2001.

- [13] Jürgen Bernard, Nils Wilhelm, Björn Krüger, Thorsten May, Tobias Schreck, and Jörn Kohlhammer. Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE transactions on visualization and computer graphics*, 2013.
- [14] Avrim Blum and Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*, pages 1006–1014, 2015.
- [15] Colin R Blyth. On simpson’s paradox and the sure-thing principle. *Journal of the American Statistical Association*, 67(338):364–366, 1972.
- [16] Carlo Emilio Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Libreria internazionale Seeber*, 1936.
- [17] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [18] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.
- [19] Gavin C Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *The 2006 ieee international joint conference on neural network proceedings*, pages 1661–1668. IEEE, 2006.
- [20] Gavin C Cawley and Nicola LC Talbot. Efficient approximate leave-one-out cross-validation for kernel logistic regression. *Machine Learning*, 71(2-3):243–264, 2008.
- [21] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.
- [22] CDC. Behavioral risk factor surveillance system questionnaire. <https://www.cdc.gov/brfss/questionnaires/index.htm>. Accessed: 2018-02-20.
- [23] Ugur Cetintemel, Mitch Cherniack, Justin DeBrabant, Yanlei Diao, Kyriaki Dimitriadou, Alexander Kalinin, Olga Papaemmanouil, and Stanley B Zdonik. Query steering for interactive data exploration. In *CIDR*, 2013.
- [24] Andrew Crotty, Alex Galakatos, Emanuel Zraggen, Carsten Binnig, and Tim Kraska. Vizdom: Interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8(12):2024–2027, 2015.
- [25] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 517–528. ACM, 2014.
- [26] Kevin Donnelly. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 2006.

- [27] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, pages 2350–2358, 2015.
- [28] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.
- [29] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126, 2015.
- [30] Chuan-sheng Foo, Chuong B Do, and Andrew Y Ng. Efficient multiple hyperparameter learning for log-linear models. In *Advances in neural information processing systems*, pages 377–384, 2008.
- [31] Dean P Foster and Robert A Stine. α -investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society: Series B*, 2008.
- [32] Andrew Gelman, Jennifer Hill, and Masanao Yajima. Why we (usually) don’t have to worry about multiple comparisons. *Journal of Research on Educational Effectiveness*, 2012.
- [33] Andrew Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no “fishing expedition” or “p-hacking” and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University*, 2013.
- [34] Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.
- [35] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.
- [36] Peter Hall and Andrew P Robinson. Reducing variability of crossvalidation for smoothing-parameter choice. *Biometrika*, 96(1):175–186, 2009.
- [37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
- [38] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.
- [39] Ruth Heller, Damian Stanley, Daniel Yekutieli, Nava Rubin, and Yoav Benjamini. Cluster-based analysis of fmri data. *NeuroImage*, 33, 2006.
- [40] Yosef Hochberg. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 1988.

- [41] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 1979.
- [42] James X Hu, Hongyu Zhao, and Harrison H Zhou. False discovery rate control with groups. *Journal of the American Statistical Association*, 2010.
- [43] Stratos Idreos and Erietta Liarou. dbtouch: Analytics at your fingertips. In *CIDR*, 2013.
- [44] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. 2015.
- [45] John PA Ioannidis. Why most published research findings are false. *PLoS medicine*, 2005.
- [46] Yannis Katsis, Natasha Balac, Derek Chapman, Madhur Kapoor, Jessica Block, William G. Griswold, Jeannie Huang, Nikos Koulouris, Massimiliano Menarini, Viswanath Nandigam, Mandy Ngo, Kian Win Ong, Yannis Papakonstantinou, Besa Smith, Konstantinos Zarifis, Steven Woolf, and Kevin Patrick. Big data techniques for public health: A case study. In *Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE/ACM, 2017.
- [47] Yannis Katsis, Nikos Koulouris, Yannis Papakonstantinou, and Kevin Patrick. Assisting discovery in public health. *HILDA workshop, ACM SIGMOD*, 2017.
- [48] Douglas B Kell and Stephen G Oliver. Here is the evidence, now what is the hypothesis? the complementary roles of inductive and hypothesis-driven science in the post-genomic era. *Bioessays*, 26(1):99–105, 2004.
- [49] Alicia Key, Bill Howe, Daniel Perry, and Cecilia Aragon. Vizdeck: self-organizing dashboards for visual analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 681–684. ACM, 2012.
- [50] Carol Kilkenny, William J Browne, Innes C Cuthill, Michael Emerson, and Douglas G Altman. Improving bioscience research reporting: the arrive guidelines for reporting animal research. *PLoS biology*, 2010.
- [51] Josef Kittler. Feature set search algorithms. *Pattern recognition and signal processing*, 1978.
- [52] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.
- [53] John Langford. Clever methods of overfitting. <https://hunch.net/?p=22>. Accessed: 2020-02-20.
- [54] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

- [55] J Loughrey and P Cunningham. Using early-stopping to avoid overfitting in wrapper-based feature subset selection employing stochastic search. In *10th UK Workshop on Case-Based Reasoning*, pages 3–10, 2005.
- [56] John Loughrey and Pádraig Cunningham. Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 33–43. Springer, 2004.
- [57] Rupert G Miller. Normal univariate techniques. In *Simultaneous statistical inference*, pages 37–108. Springer, 1981.
- [58] Tom M Mitchell. *Machine learning*. McGraw-hill New York, 1997.
- [59] Marcus R Munafò, Brian A Nosek, Dorothy VM Bishop, Katherine S Button, Christopher D Chambers, Nathalie Percie Du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J Ware, and John PA Ioannidis. A manifesto for reproducible science. *Nature human behaviour*, 1(1):1–9, 2017.
- [60] Arnab Nandi, Lilong Jiang, and Michael Mandel. Gestural query specification. *Proceedings of the VLDB Endowment*, 7(4):289–300, 2013.
- [61] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.
- [62] Kobbi Nissim and Uri Stemmer. On the generalization properties of differential privacy. *arXiv preprint arXiv:1504.05800*, 2015.
- [63] Oluwakemi Ola and Kamran Sedig. The challenge of big data in public health: an opportunity for visual analytics. *Journal of public health informatics*, 2014.
- [64] Yuan Qi, Thomas P Minka, Rosalind W Picard, and Zoubin Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings of the twenty-first international conference on Machine learning*, page 85, 2004.
- [65] R Bharat Rao, Glenn Fung, and Romer Rosales. On the dangers of cross-validation. an experimental evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 588–596. SIAM, 2008.
- [66] Anat Reiner, Daniel Yekutieli, and Yoav Benjamini. Identifying differentially expressed genes using false discovery rate controlling procedures. *Bioinformatics*, 2003.
- [67] Juha Reunanen. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3(Mar):1371–1382, 2003.
- [68] Joseph P Romano, Azeem M Shaikh, and Michael Wolf. Control of the false discovery rate under dependence using the bootstrap and subsampling. *Test*, 2008.

- [69] Mauricio Santillana, André T Nguyen, Mark Dredze, Michael J Paul, Elaine O Nsoesie, and John S Brownstein. Combining search, social media, and traditional data sources to improve influenza surveillance. *PLoS computational biology*, 2015.
- [70] Mikhail M Savitski, Mathias Wilhelm, Hannes Hahne, Bernhard Kuster, and Marcus Bantscheff. A scalable approach for protein false discovery rate estimation in large proteomic datasets. *Molecular & Cellular Proteomics*, 2015.
- [71] Thibault Sellam, Robin Cijvat, Richard Koopmanschap, and Martin Kersten. Blaeu: mapping and navigating large tables with cluster analysis. *VLDB*, 2016.
- [72] Juliet Popper Shaffer. Multiple hypothesis testing. *Annual review of psychology*, 1995.
- [73] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- [74] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *VLDB*, 2016.
- [75] R John Simes. An improved bonferroni procedure for multiple tests of significance. *Biometrika*, 1986.
- [76] Richard C Sprinthal. *Basic statistical analysis*. Allyn & Bacon, 2003.
- [77] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 2002.
- [78] Lei Sun, Radu V Craiu, Andrew D Paterson, and Shelley B Bull. Stratified false discovery control for large-scale hypothesis testing with application to genome-wide association studies. *Genetic Epidemiology*, 2006.
- [79] Tableau. Tableau, 2018. Accessed: 2018-05-05.
- [80] Michelle Takemoto, Jordan A Carlson, Kevin Moran, Suneeta Godbole, Katie Crist, and Jacqueline Kerr. Relationship between objectively measured transportation behaviors and health characteristics in older adults. *International journal of environmental research and public health*, 2015.
- [81] Tim van der Zee, Jordan Anaya, and Nicholas JL Brown. Statistical heartburn: an attempt to digest four pizza publications from the cornell food and brand lab. 2017.
- [82] Manasi Vartak, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. Seedb: automatically generating query visualizations. *VLDB*, 2014.
- [83] Brian Wansink. The grad student who never said no, 2017. Accessed: 2018-02-28.

- [84] Christopher KI Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.
- [85] Eugene Wu, Leilani Battle, and Samuel R Madden. The case for data visualization management systems. *VLDB*, 2014.
- [86] Yifan Wu, Larry Xu, Remco Chang, and Eugene Wu. Towards a bayesian model of data visualization cognition, 2017.
- [87] Sonja Zehetmayer and Martin Posch. False discovery rate control in two-stage designs. *BMC bioinformatics*, 13(1):81, 2012.
- [88] Zheguang Zhao, Lorenzo De Stefani, Emanuel Zraggen, Carsten Binnig, Eli Upfal, and Tim Kraska. Controlling false discoveries during interactive data exploration. *SIGMOD*, 2017.
- [89] Alain F Zuur, Elena N Ieno, and Chris S Elphick. A protocol for data exploration to avoid common statistical problems. *Methods in ecology and evolution*, 2010.