OUTSIDE THE MACHINE LEARNING BLACKBOX: SUPPORTING ANALYSTS BEFORE AND AFTER THE LEARNING ALGORITHM

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Miles Arthur Munson

May 2010

OUTSIDE THE MACHINE LEARNING BLACKBOX: SUPPORTING ANALYSTS BEFORE AND AFTER THE LEARNING ALGORITHM

Miles Arthur Munson, Ph.D.

Cornell University 2010

Applying machine learning to real problems is non-trivial because many important steps are needed to prepare for learning and to interpret the results after learning. This dissertation investigates four problems that arise before and after applying learning algorithms.

First, how can we verify a dataset contains "good" information? I propose *cross-data validation* for quantifying the quality of a dataset relative to a benchmark dataset and define a *data efficiency ratio* that measures how efficiently the dataset in question collects information (relative to the benchmark). Using these methods I demonstrate the quality of bird observations collected by the eBird citizen science project which has few quality controls.

Second, can off-the-shelf algorithms learn a model with good task-specific performance, or must the user have expertise both in the domain and in machine learning? In many applications, standard performance metrics are inappropriate, and most analysts lack the expertise or time to customize algorithms to optimize task-specific metrics. Ensemble selection offers a potential solution: build an ensemble to optimize the desired metric. I evaluate ensemble selection's ability to optimize for domain-specific metrics on natural language processing tasks and show that ensemble selection usually improves performance but sometimes overfits.

Third, how can we understand complex models? Understanding a model often is as important its accuracy. I propose and evaluate statistics for measuring the importance of inputs used by a decision tree ensemble. The statistics agree

with sensitivity analysis and, in an application to bird distribution models, are 500 times faster to compute. The statistics have been used to study hundreds of bird distribution models.

Fourth, how should data be pre-processed when learning a high-performing ensemble? I examine the behavior of variable selection and bagging using a bias-variance analysis of error. The results show that the most accurate variable subset corresponds to the best bias-variance trade-off point. Often, this is *not* the point separating relevant from irrelevant inputs. Variable selection should be viewed as a variance reduction method and thus is often redundant for low variance methods like bagging. The best bagged model performance usually is obtained using all available inputs.

BIOGRAPHICAL SKETCH

Art Munson was born in Ithaca, NY and grew up in Lansing, NY. During his senior year of high school he took his first programming class, beginning with how to turn on the computer. From this humble start Art quickly became fascinated with the logic and structure of computers and with their data processing abilities.

In 2001 Art graduated from Williams College, receiving a bachelor of arts in computer science with honors. His thesis advisor, Duane Bailey, nudged him towards graduate school, but Art failed to notice the hint and instead became a software developer at Microsoft.

In 2003 Art decided that software development was not going to be fun forever and returned to Ithaca, NY to get a Ph.D. in computer science from Cornell University. At Cornell he studied machine learning and ecological modeling under the guidance of Rich Caruana.

To S. Charles Doret,

a great friend

who inspired me to get a Ph.D.

To Katy,

my wife and best friend

who supported me so that I *did* get the Ph.D.

ACKNOWLEDGMENTS

This dissertation would not exist with out the support and guidance of many people.

First and foremost, Rich Caruana has been an excellent advisor. By his example and guidance I have learned to be optimistic in the face of temporary setbacks and to be careful (paranoid?) while checking the correctness of experimental results. His insight into experimental results and the best way to communicate those results was invaluable for helping me find the forest through the trees. I cannot thank Rich enough for countless hours spent discussing every idea we could conceive. I am still amazed that, after taking a position at Microsoft, he set aside one evening a week to continue our regular meetings.

Claire Cardie was also a valuable advisor at the beginning and end of my Ph.D. work. I began my Ph.D. research with Claire while doing research in natural language processing using machine learning methods and later transitioned to working with Rich full time to focus more fully on machine learning. After Rich went to Microsoft, Claire graciously agreed to become co-chair of my committee and provided welcome advice during my job search and while I wrote my dissertation. Claire taught me to look for the silver lining in surprising results and that scientific writing can be infitely compressed.[1]

Thanks are also owed to the other members of my special committee, Ken Birman and Michael Spivey. Ken supported and encouraged me in his unique way; my favorite encouragement was him telling me, "I'm not letting you leave Cornell without a Ph.D."[2] Spivey introduced me to fascinating questions about

[1]The first draft of our EMNLP/HLT paper (Chapter 4) exceeded the conference's page limit by two pages. I *thought* I had done a good job cutting excess verbiage and making the writing concise, but the second draft was still too long by a page. Claire revised the introduction and abstract and not only made them shorter but also *added* multiple important points.

[2]I guess Ken did not realize that I had zero intention of leaving, short of being kicked out.

human cognition, memory, and thinking and reminded me that learning (human or machine) is a remarkable process.

I have been blessed with many wonderful collaborators both from Cornell's computer science department and from the Cornell Lab of Ornithology. From computer science I thank Alex Niculescu-Mizil, Daria Sorokina, Dan Sheldon, Mirek Riedewald, and Mohamed Elhawary. Alex in particular influenced and shaped my approach to conducting empirical research.[3] From the Lab of Ornithology I thank Wesley Hochachka, Steve Kelling, Daniel Fink, Kevin Webb, Ken Rosenberg, Brian Sullivan, Marshall Iliff, and Chris Wood. Footnotes at the beginning of Chapters 3–6 acknowledge the specific co-authors for each piece of research. In addition to my gratitude for successful collaborations, I thank the team from ornithology for giving me a second academic home and for inspiring in me a new-found interest in birds and the natural world in general. During the summer of 2007 I had the pleasure of working with Misha Bilenko at Microsoft Research on how to improve web search. Misha's enthusiasm and energy are contagious, and they made my internship pass too quickly.

The research in this dissertation was generously funded by the Advanced Research and Development Activity, the Leon Levy Foundation, the Wolf Creek Foundation, and NSF grants EF-0427914, IIS-0208028, IIS-0412930, IIS-0612031, IIS-074826, IIS-0832782, ITR-0427914, DBI-0542868, DUE-0734857. Thanks to this support I have been a well-fed graduate student instead of a starving graduate student.

One of the best parts of my time at Cornell has been meeting many good friends. My fellow graduate students fed my intellectual curiousity with many interesting discussions about research and life, gave feedback on paper drafts and practice talks, and helped sustain my mental health through strategic pro-

[3] Specifically, I blame Alex for teaching me that it is *never* too late to run another experiment.

crastination and recreation. My graduate schooling has been long, and the list of friends is simlarly long. I thank in particular Selcuk Aya, Mahesh Balakrishnan, Eric Breck, Jany Chan, Yejin Choi, Thomas Finley, Nikos Karampatziakis, Fang Liu, Tudor Marian, Filip Radlinski,[4] Vasu Raman, Benyah Shaparenko, Dan Sheldon, Daria Sorokina, Vidhya Venkataraman, Dan Williams, Ainur Yessenalina, Yisong Yue, and Katie Zobeck.[5]

Thank you to all the administrative assistants at the computer science department for making the bureaucracy and logistics needed for graduate school as painless as possible so I could focus on research. In particular, Becky Stewart, Stephanie Meik, and Melissa Totman patiently answered all my questions.

My family have given me support, love, and the occasional reminder to hurry up and finish already. Thank you Mom, Dad, Terri, David, Paul, and Ross.

And to my wife Katy: thank you for your tireless support, for patiently ignoring my bad moods when research did not go well, for putting up with 24-hour deadline pushes, for a constant supply of delicious home cooked food, for reminding me that life is more than just work, and for, well, everything. More than anyone, you helped me to not only reach the finish line but to reach it with my sanity intact.

[4]Special thanks to Filip for procrastinating during his dissertation writing to help me add insulation to my house in October 2007.

[5]Tudor Marian, Dan Williams, and Benyah Shaparenko deserve special mention for helping me re-roof my house on the hottest weekend of summer 2009.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

> *Researchers are using many different methods to collect or generate data—from sensors and CCDs to supercomputers and particle colliders. When the data finally shows up in your computer, what do you do with all this information that is now in your digital shoebox?*
>
> — Jim Gray, 2007 (Hey *et al.*, 2009, p. xviii)

We are in the middle of a data revolution. Unprecedented volumes of data are overwhelming scientists in fields as diverse as astronomy,[1] atmospheric science,[2] biology,[3] ecology (Wilson, 2003; Sullivan *et al.*, 2009; Hunt *et al.*, 2009), environmental science,[4] medicine (Buchan *et al.*, 2009; Gillam *et al.*, 2009), oceanography,[5] and sociology.[6] Visualizing, analyzing, and understanding so much data requires new approaches and tools. Machine learning, with its ability to extract information from data, is an important toolkit for working in a data-rich world. Despite the availability of sophisticated learning algorithms that can detect and leverage complex patterns in data, machine learning is not widely used to solve problems. Adoption is slow because applying machine learning methods requires a great deal of machine learning expertise. Important practical considerations like how to pose learning problems, prepare data, understand learned models, and get confidence assessments for predictions have either been largely

[1] Large-scale data collection projects in astronomy include the Allen Telescope Array, the Sloan Digital Sky Survey (Abazajian *et al.*, 2009), Pan-STARRS, and the forthcoming Square Kilometer Array. Through the Galaxy Zoo game citizen scientists are analyzing this massive data and have already a) refuted the claim that galaxies preferentially rotate in one direction, and b) found the bluest known galaxy in the universe (Goodman & Wong, 2009).

[2] `http://www.ncar.ucar.edu/tools/datasets/`

[3] Entrez, the Life Sciences Search Engine, provides access to 29 biology and chemistry related data sets (e.g., nucleotide sequences, protein sequence, 3D protein structures, and chemical substances).

[4] Data Observation Network for Earth: `http://dataone.org`

[5] Ocean Observatories Initiative: `http://www.oceanleadership.org/programs-and-partnerships/ocean-observing/ooi/`

[6] `http://www.facebook.com`

ignored or have received much less attention than the development of new learning algorithms. These are serious obstacles for scientists, analysts, and companies. This dissertation studies four practical problems that arise before and after using machine learning to learn a model from data.

1.1 The Modeling Pipeline

There are many steps required to learn a useful model from data, and only one of them actually learns the model. Figure 1.1 depicts the pipeline, or sequence, of steps. There are two inputs to this process: data, and domain knowledge that is already known. After the data is pre-processed, a learning algorithm combines the data and pre-existing domain knowledge to learn (construct) a model that explains the data. For example, one model might be a series of rules describing bird migration paths; the paths could depend on the bird species, weather, time of year, and geographic features. A model may serve different purposes, including making predictions on future data (e.g., given last night's weather, are red-tailed hawks likely to migrate today?) and knowledge discovery (e.g., what are the rules that accurately describe migration?).

1.2 Why Studying Bird Populations is Important and Hard

This dissertation investigates practical modeling problems primarily in the context of ecological modeling, although one problem is studied in the context of natural language processing. The specific ecological domain studied is how to model populations of bird species using **citizen science data** (data collected by volunteers (Bonney *et al.*, 2009)).

Accurate models of avian populations have practical importance. First, birds

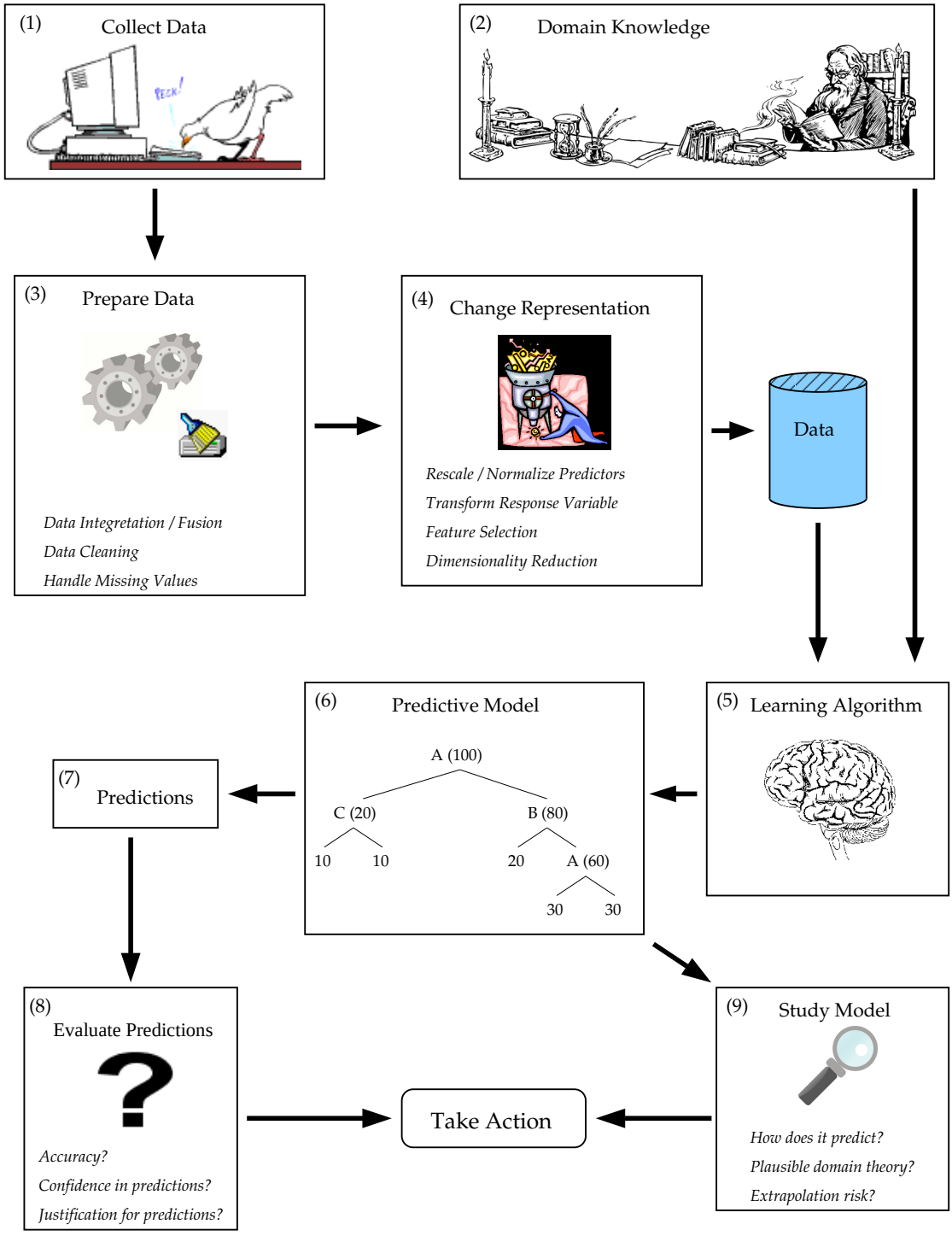

Figure 1.1: The modeling pipeline. Information flows from the inputs (data and domain knowledge) to a learned model and on to the data analyst.

are important **bioindicators**, i.e., they provide information about the health of the environment (Markert *et al.*, 2003). The status of avian populations is an important sentinel that can warn about ecosystem disruptions that would otherwise go unnoticed.[7] Second, accurate population models help inform conservation and land management efforts (e.g., Young *et al.*, 2009). Third, knowledge of avian populations, particularly migration patterns, is important for understanding and predicting the spread of diseases like avian influenza (Kilpatrick *et al.*, 2006).

Avian population modeling poses many challenges, including:

- Birds populations are dynamic and in constant flux. In addition to long distance seasonal migrations, individual birds make local-scale movements daily (e.g., to forage for food). For birds with large territories (e.g., great blue herons), local movements can span miles.
- Many intercorrelated environmental factors underlie population dynamics. Climate patterns, elevation, available habitat and habitat configurations, food and water availability, proximity to human populations, and weather can all effect the distribution of a bird species. Some important factors (e.g., food and water availability) are hard to directly measure except in small scale studies.
- Large-scale modeling requires large-scale collection of bird observations that is accomplished by networks of volunteer observers with wide ranges of skill. In general, the greater the volume of volunteer data collected, the less control can be imposed on how the data is collected.

[7] For example, raptor and seabird populations plummeted in the 1950's and 1960's in industrialized countries and helped raised awareness of problems caused by pesticides. Subsequent study and understanding of the detrimental effect of these compounds led to laws restricting or banning their use by the 1970's. Since then, raptor and seabird populations have begun to recover, providing valuable evidence that ecosystems are recovering as a result of pesticide regulation (Becker, 2003).

- Even with large networks of observers collecting data, observations across space and time are sparse when one seeks to model populations year-round at continent-wide scales. Relatively small sample sizes increase the apparent randomness of bird counts.
- Detecting and identifying birds is a noisy process. Even experts counting birds at the same time and locations will record different species lists and counts (Faanes & Bystrak, 1981). This is due to a) limits in how many observations a single observer can simultaneously make, b) varying skill levels of observers, and c) the fact that a bird can be present in an area but impossible to detect (because it is hidden and not vocalizing). Environmental factors like habitat and weather can also impact the detection process (Jobin *et al.*, 1996; Robbins, 1981a).

Overcoming these challenges to obtain a clear *aggregate* picture of avian populations requires sophisticated methods that can automatically learn models from sparsely sampled and noisy data.

1.3 Problems Studied

1.3.1 Quantifying Relative Data Quality

The first problem studied in this dissertation is how to verify that a data source contains sufficient "good" information to be used for modeling (part of box 3 in Figure 1.1). Fundamentally, one needs good data to build a good model. Data quality has traditionally been guaranteed through careful, controlled data collection that is designed to control for confounding effects and limit bias and noise. As data collection increases in scale, however, data quality is harder to

ensure. This is particularly true for large-scale observational data where little (or no) effort is made to control for confounding effects.

For example, the eBird project (Sullivan *et al.*, 2009) collects hundreds of thousands of checklists from volunteers who record which bird species they found while birding. The sheer volume of this data, and the fact that it is collected year-round throughout the western hemisphere, could make eBird a valuable source of information for monitoring the status and distribution of bird species. Yet it is also possible that biases in the data collection (e.g., most checklists are collected near population centers) and uncontrolled sources of variance (e.g., birding trips last variable time lengths; birder expertise varies) hide the underlying biological information. How can we evaluate the utility and effectiveness of such a data source? Given a choice between this data source and another, which one should be used for model building?

Chapter 3 describes *cross-data validation*, a framework for quantifying the quality of a dataset relative to a benchmark dataset whose quality is known *a priori*. I define a *data efficiency ratio* that measures how efficiently the dataset in question collects information (again, relative to the benchmark data). Using these methods, I show that eBird data contain information similar in quality to that in data from the highly standardized North American Breeding Bird Survey (Robbins *et al.*, 1986), while the information per breeding bird survey datum is higher.

1.3.2 Optimizing for Arbitrary Performance Measures

The second problem studied is how to learn a model with good task-specific performance using off-the-shelf machine learning algorithms (part of boxes 5 and 8 in Figure 1.1). Machine learning algorithms typically optimize for good

accuracy, log-likelihood, squared error, or hinge loss. For many applications, however, these standard performance measures are inappropriate. For example, natural language processing (NLP) data can be highly skewed in its distribution of positive and negative examples, and a measure that focuses on the performance of the minority cases is more appropriate (e.g., F-measure). As a second example, result rankings from web search engines are often measured using normalized discounted cumulative gain (NDCG) (Järvelin & Kekäläinen, 2002), a metric that weights the quality of top ranked results more heavily than low ranked results. While learning algorithms can be custom designed to optimize for task-specific performance measures like F-measure and NDCG, this requires time and considerable machine learning expertise that most data analysts do not have. Existing learning algorithms with readily available implementations are the only realistic options for most machine learning practitioners.

For standard performance metrics like accuracy and squared error, ensemble learning methods are among the highest performing methods currently available. **Ensemble methods** build accurate models by combining the predictions of base models into a committee of models called an **ensemble**. (Section 2.6 describes ensemble learning in more detail.) Applying ensemble learning methods to a modeling task is straightforward if one is able to build the constituent base models. In most cases, base models can be learned using off-the-shelf learning algorithms.

Chapter 4 studies an opportunity enabled by ensemble learning: incrementally building an ensemble model to optimize for any domain-specific performance measure. Specifically, chapter 4 evaluates whether a method called ensemble selection can successfully optimize for domain-specific performance metrics on natural language processing tasks. **Ensemble selection** (Caruana

et al., 2004) builds an ensemble by incrementally selecting trained models from a large heterogeneous model library. Models are selected to optimize an arbitrary performance criterion. The model library contains a large number of models trained using a variety of standard learning algorithms and parameter settings. Using three natural language processing tasks as examples, I show that:

1. Exploring multiple algorithms and parameters is very important for getting good performance. Comparing algorithms without parameter tuning (common practice in the NLP community when I conducted the evaluation) can result in false conclusions.
2. Ensemble selection can optimize for unusual performance metrics, but the library needs to contain enough models that perform well for the desired metric.

1.3.3 Understanding Decision Tree Ensembles

The third problem studied is how to understand complex models (part of box 9 in Figure 1.1). For many applications, understanding how the model makes predictions is at least as important as the accuracy of the predictions. For example, in a collaboration with the Cornell Lab of Ornithology, we trained ensembles of decision trees to accurately predict the distribution of bird populations. (Section 2.5 reviews decision trees.) The biologists were more interested in how the models made accurate predictions than the predictions themselves.

While a single decision tree is a relatively transparent model that can be studied and understood, an ensemble of decision trees is opaque and difficult to understand. This is the main drawback to ensemble learning: while combining models produces more accurate predictions, that same combination process makes an ensemble virtually incomprehensible.

An important first step in understanding a model is to identify which model inputs are important. In other words, which inputs strongly impact the model's predictions? The importance of an input can be measured using **sensitivity analysis**: perturbing the input (e.g., replacing the input values with noise) and observing how much the perturbation hurts model performance. Sensitivity analysis can be applied to any model, but it requires perturbing each input separately. Consequently, sensitivity analysis is computationally expensive for models that use hundreds or thousands of inputs.

Chapter 5 proposes statistics for measuring the importance of model inputs that can be efficiently computed from the structure of decision tree ensemble models. These statistics correlate well with sensitivity analysis and are 500 times faster to compute for the bird distribution models. Once important inputs are identified, they can be visualized using partial dependence functions (Friedman, 2001). The statistics were used to study distribution models for dozens of species, in 27 regions across the United States. The results are available online[8] for biologists to explore the patterns of birds during the winter.

1.3.4 Interaction Between Variable Selection and Bagging

The fourth problem studied is how to best pre-process data, particularly for learning a high-performing ensemble (part of box 4 in Figure 1.1). **Variable selection** is an important pre-processing tool that seeks to identify and keep only the important inputs (also called **predictor variables**, or simply **predictors**). In addition to simplifying models, variable selection will often also improve model accuracy by getting rid of distracting inputs. Surprisingly, when we applied variable selection to the problem of modeling bird distributions, we found that

[8] `http://www.avianknowledge.net/content/toolbox/partial-dependency-plots/`

removing seemingly unimportant variables consistently hurt the performance of bagged models (a particular kind of ensemble model). Is this interaction between bagging and variable selection a general pattern, or is it a peculiarity of modeling species distributions? If it is general, why?

Chapter 6 examines the behavior of variable selection and bagging using a bias/variance analysis of error. (The bias-variance decomposition of error will be explained in the next chapter.) The results show that the most accurate predictor subset corresponds to the best bias-variance trade-off point for the learning algorithm. Often, this is not the point separating relevant from irrelevant predictor variables, but where increasing variance outweighs the gains from adding more (weakly) relevant predictors. In other words, variable selection can be viewed as a variance reduction method that trades off the benefits of decreased variance (from the reduction in input dimensionality) with the harm of increased bias (from eliminating some of the relevant predictors). If a variance reduction method like bagging is used, more (weakly) relevant predictors can be exploited and the most accurate variable subset is usually larger. In many cases, the best performance for bagged models is obtained by using all available predictors.

1.4 The Importance and Difficulty of Modeling Steps

Before continuing on to the research results, let us briefly revisit this chapter's opening claims that 1) the pre- and post-processing steps surrounding model building are non-trivial, and that 2) these practical considerations have been largely ignored or under studied by the research community. To explore the first claim, I surveyed machine learning and data mining practitioners about what fraction of time is spent in each of the modeling stages in Figure 1.1. (Survey

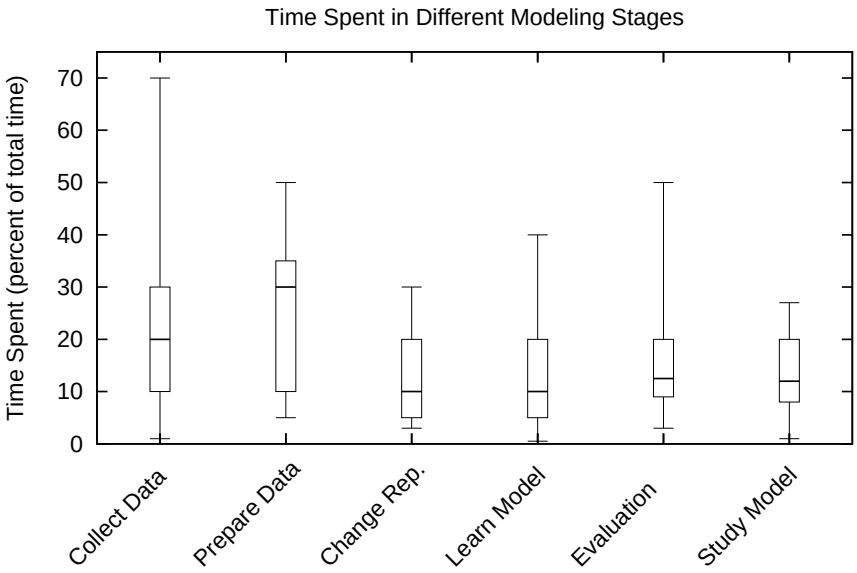

Figure 1.2: Allocation of time spent building systems with machine learning or data mining components. Time estimates were collected from practitioners with experience deploying one or more systems. Boxes show the 25th and 75th quantiles of time spent per stage; the line within each box marks the median time spent. Whiskers show the minimum and maximum time spent. See Appendix A for survey details.

details are in Appendix A.) The relative time spent in each stage varied greatly by project (Figure 1.2). Data collection and preparation were the most time consuming stages, based on median values (20% and 30%, respectively). Median times spent on other stages were all around 10%. *In other words, the stages that precede and follow model building are* individually *at least as time consuming as learning the model. In the typical project, only 10% of the effort is actually spent learning the model* (Figure 1.2). Furthermore, most survey respondents also felt that all modeling steps were important for building successful systems (Figure 1.3).

I investigated the second claim in two ways. First, the survey described above also asked how much energy the research community spent on each step. Respondents felt that the community spends the most energy on learning algorithms (Figure 1.4). Second, I manually categorized the papers published at the *International Machine Learning Conference* and the *ACM SIGKDD Conference*

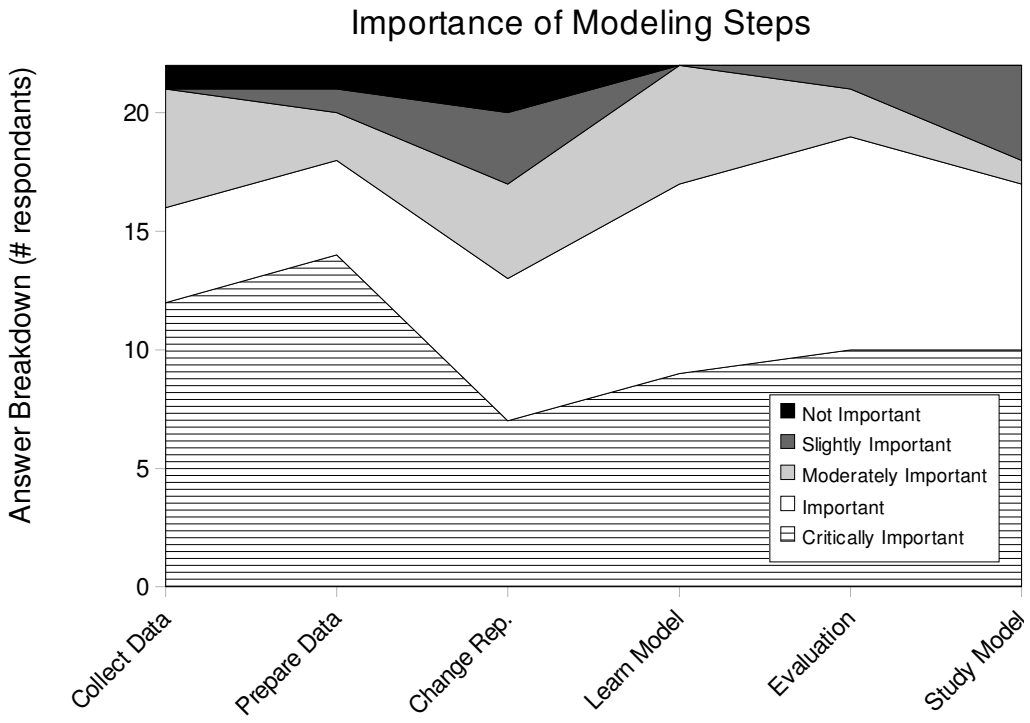

Figure 1.3: The majority of surveyed practitioners rated all steps as important or critically important to their systems' successes. Chart shows the breakdown of importance ratings for each modeling step. For example, of the 22 total respondents, 12 rated data collection *critically important*, 4 rated it *important*, 5 rated it *moderately important*, and 1 rated it *not important*. See Appendix A for survey details.

on Knowledge Discovery and Data Mining in 2009, the top machine learning and data mining conferences, respectively. I labeled each paper as addressing one or more of the modeling stages from Figure 1.1; an extra category, *Other*, was added to catch papers that did not fit easily into the modeling stage categories. The distribution of papers at these conferences supports the beliefs of survey respondents: the research community spends most of its energy on learning algorithms (Figure 1.5). While the majority of respondents felt that all stages except data collection received at least moderate research energy (Figure 1.4), the distribution of papers suggests a more skewed focus, particularly at ICML. Significant effort is being spent on how to change representations to improve

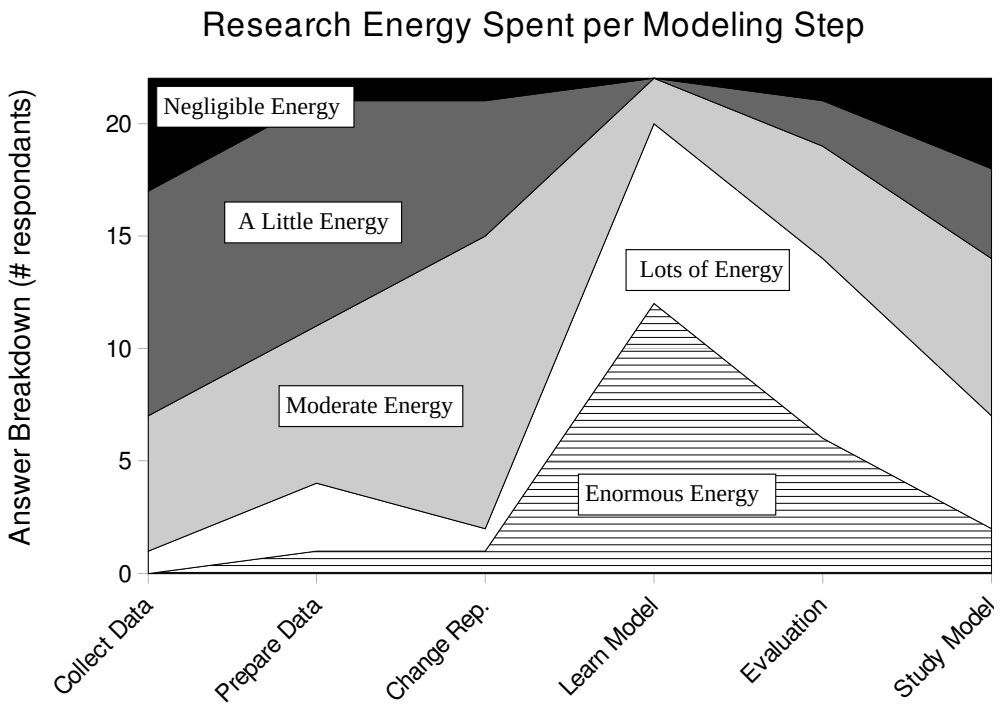

Figure 1.4: Most surveyed practitioners felt that the machine learning and data mining research communities spend the most energy on how to learn a model from data. Most respondents rated the communities as spending *moderate energy* or less on the modeling steps preceding and following learning a model. In contrast 19 of 22 respondents felt the community spent *lots of energy* or *enormous energy* on how to learn a model. See Appendix A for survey details.

learning, but minimal amounts of research address the remaining steps in the modeling pipeline (Figure 1.5). The distribution of papers at KDD was slightly more balanced but the focus was still clearly on learning algorithms. A significant number of KDD papers were categorized as Other; 3/4 of these papers were case studies of applying machine learning and data mining to solve real problems.

Admittedly, the survey results and paper categorizations are far from rigorous. First, there is no way to know if the survey collected a representative sample of practitioners. Second, the survey sample size is small, with only 22 respondents. Third, the distribution of papers in recent conferences is based

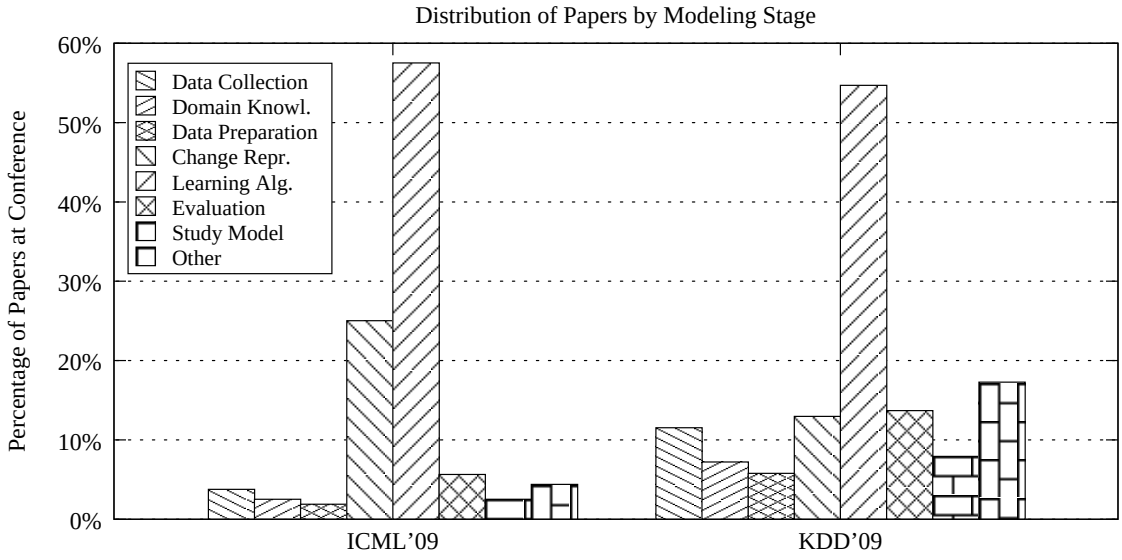

Figure 1.5: Distribution of papers published at ICML 2009 and KDD 2009. Papers were manually categorized as addressing one or more of the boxes in Figure 1.1. Percentages do not add to 100 because some papers were counted in multiple categories.

on my own subjective judgment and cursory reviews of 299 papers. Fourth, practitioners may have a skewed perspective of how researchers spend their energy. Nonetheless, the differences between where researchers focus their energy and where practitioners spend their energy are so striking that it is hard to completely dismiss this evidence.

1.5 Summary

In summary, there is tremendous room for simplifying the important, practical steps that precede and follow the actual model learning. This dissertation contains a few small steps towards improving this situation. In the next chapter I present background that is fundamental to this work. Chapters 3–6 describe the four problems studied in more detail and present the research results. Chapter 7 summarizes the dissertation's conclusions and briefly discusses future work.

CHAPTER 2

BACKGROUND

Let us now suppose that in the mind of each man there is an aviary of all sorts of birds — some flocking together apart from the rest, others in small groups, others solitary, flying anywhere and everywhere. . . . We may suppose that the birds are kinds of knowledge, and that when we were children, this receptacle was empty; whenever a man has gotten and detained in the enclosure a kind of knowledge, he may be said to have learned or discovered the thing which is the subject of the knowledge: and this is to know.

— Plato, *The Republic* (Bartlett & Kaplan, 1992)

This chapter presents the background material for the dissertation and describes prior work. After defining some terminology (§ 2.1) and notation (§ 2.2), I define performance measures used throughout the dissertation (§ 2.3). I then review the statistical notions of bias, variance, and noise and explain how these concepts relate to data quality and the performance of learning algorithms (§ 2.4). Sections 2.5 and 2.6 review the primary learning methods used in the dissertation: decision trees and ensemble learning methods. In Section 2.7 I describe the variable selection problem and the two classic variable selection algorithms used in Chapter 6. Section 2.8 describes the task that drives most of this research: modeling bird species distribution.

2.1 Terminology

This dissertation focuses on **predictive models**: models that can make predictions about future data. This is not much of a restriction because most models, in some way, can be used to make predictions. Some kinds of models, however, are purely descriptive: they describe and summarize their input data but cannot generalize to answer questions about future data. For example, hierarchical

clustering (Hand *et al.*, 2001) produces a model, called a dendrogram, that describes how similar each data point is to the others and how the data cluster into groups. The dendrogram does not contain any information about future data points.

The kind of machine learning used to build models in this dissertation is called **supervised machine learning**. In **supervised machine learning** the computer is given examples of the data we want modeled and the correct answer for each example. In other words, each **example** is a pair consisting of a data record (the model inputs) and the desired model output for that record (often called the **label**). The set of examples is called the **training set** or **training data**. The computer runs a supervised learning algorithm to construct a model of the data that (hopefully) predicts the outputs given the inputs. A model's accuracy is measured by testing its predictions on **test data**: examples not used during training. A third dataset called the **validation set** is often used to evaluate accuracy during model development and to tune the learning algorithm's hyper-parameters (defined below).

Different communities use different names for model inputs. The machine learning community typically calls inputs **features**, while the data mining community prefers the term **attributes**. The statistics community calls inputs **predictor variables** or simply **predictors**. I use these terms interchangeably, but use *predictors* in most places. Similarly, I borrow the term **response variable**, or simply **response**, from the statistics community to refer to the value that we want the model to predict using the predictors.

This dissertation only considers modeling tasks in which the response is scalar. In general, a scalar response variable can be continuous or discrete. The response variables for the problems studied in later chapters are all discrete with

a finite number of possible values. For example, the response variable for occurrence modeling (described in § 2.8) can take on one of two values: *present* or *absent*. Each allowed value for a finite discrete response is called a **class** because the modeling goal is to **classify** each example into the correct category. **Classification** is the process of learning a model to classify data. When the response is continuous or discrete but infinite, the process is called **regression**. All of the modeling problems in the following chapters are classification tasks.

I define **model learning** to include two related tasks. First, learning a model chooses all of the model's *parameter values* (often called model fitting in statistics). Further, most machine learning algorithms also choose the *structure* of the model from a large (perhaps infinite) set of possible structures. This often includes explicitly or implicitly selecting which predictors are important to include in the model. The set of structures that can be chosen is determined by the choice of learning algorithm, by domain knowledge (in some cases), and by parameter values for the learning algorithm. Note that **model parameters** and **learning algorithm parameters** are different and distinct. The former are values that complete the definition of a model and are used to make predictions. The latter control the steps used to learn the model; once the model is learned (built), they have no further impact. Learning algorithm parameters are sometimes called **hyper-parameters**. I will drop the *model* and *algorithm* modifiers and simply use *parameters* when the meaning is clear from context.

2.2 Notation

It will be useful to define some common notation. Let $E[\cdot]$ denote the expected value (i.e., mean) of the quantity inside the square brackets. Capital letters are used to denote random variables (e.g., X), while lowercase letters denote a par-

ticular sample drawn from a random variable (e.g., x drawn from the distribution X). Bold-faced letters denote vector quantities, both for samples and random variables (e.g., $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ drawn from the multivariate distribution $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$).

2.3 Performance Metrics

This section defines three performance metrics that are used in subsequent chapters to measure model performance: accuracy, mean squared error, and area under the ROC curve. Many other performance metrics are defined and used in Chapter 4.

The definitions below assume that a model predicts a probability distribution for an example, indicating the likelihood of the example belonging to each class. When the model needs to pick a single class (i.e. for accuracy), the class with the largest probability is chosen. For tasks with two classes, this is equivalent to using a threshold of 0.5.

Accuracy (ACC) is the percentage of predictions that predict the correct class label. This metric is often converted to **classification error** which equals $1 -$ accuracy. Another name for classification error is **zero-one loss** (zero loss for a correct prediction, and a loss of 1 for an incorrect prediction).

Mean squared error (MSE) is the average squared difference between the true prediction and the model's prediction. Let $\boldsymbol{x}$ denote an example, and let $p(y = k|\boldsymbol{x})$ and $q(y = k|\boldsymbol{x})$ be the true and predicted probability, respectively, that $\boldsymbol{x}$ is

class k. Then:

$$\text{MSE} \equiv \frac{1}{nK} \sum_{\boldsymbol{x}} \sum_{k} (p(y = k|\boldsymbol{x}) - q(y = k|\boldsymbol{x}))^2$$

where n is the number of examples used in the average and K is the number of classes for the task.[1] To view the error on the same scale as the predictions, RMS (root mean squared error: the square root of MSE) often is used.

Area under the ROC curve (AUC) measures the ability of a model to rank positive examples before negative examples (Fawcett, 2006). ROC curves and the AUC statistic that summarizes them only apply to binary classification tasks. The receiver operating characteristic (ROC) curve graphs the tradeoff between the model's true positive rate (percentage of positive examples correctly predicted) and false positive rate (percentage of negative examples predicted as positive); varying the threshold that discretizes predicted probabilities into positive and negative predictions changes both of these rates. Figure 2.1 depicts example ROC curves. The diagonal line represents random prediction; better models have curves that are closer to the top left corner.

The quality of the model can be summarized by the area under the ROC curve (AUC). Random predictions achieve a 0.5 AUC, and perfect predictions achieve an AUC of 1. One feature of ROC curves is that they do not depend on the proportions of positive and negative examples in the test set (Fawcett, 2006).

The AUC has a nice interpretation: it equals the probability that the model ranks a random positive example above a random negative example, which is also equivalent to the Wilcoxon test of ranks. The latter equivalence can be exploited to compute a standard error and confidence bounds for an AUC measurement (Hanley & McNeil, 1982). Conceptually, the Wilcoxon statistic W can

[1] Normalizing by K is not strictly necessary, but places MSE on the same scale regardless of the number of classes.

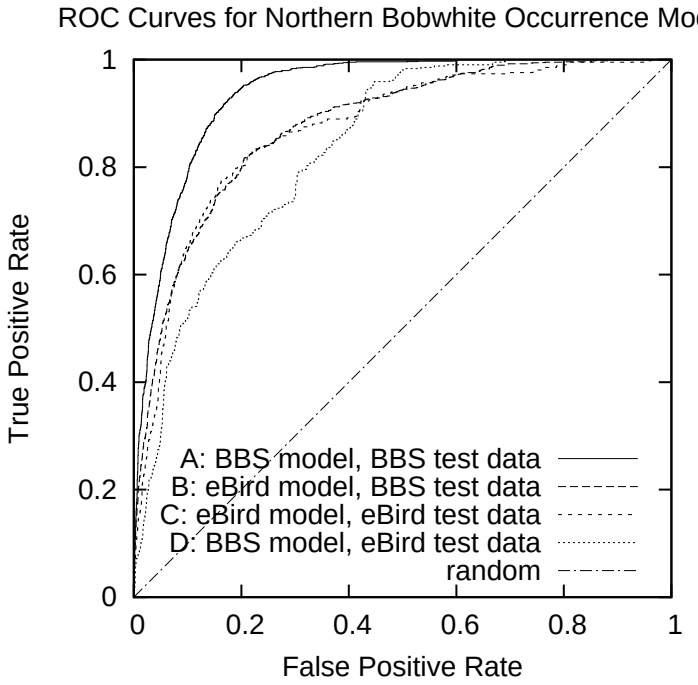

Figure 2.1: Example ROC curves. Species occurrence models for northern bobwhites were learned from two data sources—BBS and eBird—and evaluated on test data from both sources. (See Chapter 3 for why this kind of comparison is useful.) Models with curves closer to top left corner have better performance, although the best model can vary depending on the desired false positive rate. For example, curve C is preferable to curve D for most reasonable false positive rates (0 to 0.4), but curve D is preferable if high false positive rates can be tolerated.

be computed as

$$W = \frac{1}{|P||N|} \sum_{y \in P} \sum_{y' \in N} S(y, y') \tag{2.1}$$

where P and N are sets of positive and negative examples and the sums are over the model's (probability) predictions for those examples. The scoring function S is defined as:

$$S(y, y') = \begin{cases} 1 & \text{if } y > y' \\ 1/2 & \text{if } y = y' \\ 0 & \text{if } y < y' \end{cases} \tag{2.2}$$

In practice the AUC (and W) can be computed more efficiently (Fawcett, 2006), but this definition is easy to understand.

2.4 Bias, Variance, and Noise

The concepts of bias and variance appear in multiple guises in this dissertation. Statistics defines **bias** to be the difference between an estimator of a parameter value and the true parameter value θ. Let $d = d(\boldsymbol{X})$ denote the estimator for the parameter θ. The estimator estimates θ from a data sample drawn from the random variable $\boldsymbol{X}$ using the function $d(\cdot)$. Formally, the bias of d with respect to θ is

$$\text{bias}_\theta(d) = E[d(\boldsymbol{X})] - \theta \tag{2.3}$$

The expectation is taken over all possible samples drawn from $\boldsymbol{X}$ (Ross, 2004).

Variance measures the dispersion of a numeric univariate variable around its mean. The variance of a variable X is (Ross, 2004):

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2 \tag{2.4}$$

Statistics does not have a formal definition for noise. Noise is generally understood to be any unwanted signal in a measurement.

2.4.1 Biased and Noisy Data

Bias and noise are often used to describe data quality. For clarity I define what I mean by these terms in the context of data quality.[2] Let Y be a random variable representing the quantity we would like to measure (e.g., abundance or

[2]Thank you to Dan Sheldon for discussions that helped me clarify these definitions.

occurrence of a species at a particular location and time of year). Y is stochastic because I assume there will be factors affecting the quantity that cannot be observed. Samples y drawn according to Y's distribution are **unbiased**: they reflect Y and nothing else. In many cases, however, we cannot directly observe or measure the desired quantity. Instead we observe the random variable $\widehat{Y} = f(Y, \boldsymbol{X})$, a function of the true quantity and some observation process $\boldsymbol{X} = (X_1, X_2, \ldots, X_k)$ that may itself be stochastic. The bias of $\widehat{Y}$ is

$$\text{bias}_Y(\widehat{Y}) = E[\widehat{Y} - Y] = E[\widehat{Y}] - E[Y] \tag{2.5}$$

I say that $\widehat{Y}$ is a **biased variable** if this quantity is non-zero, and that samples $\widehat{y}$ drawn from $\widehat{Y}$ are **biased data**. The observation process may also increase the variance of measurements, creating a **noisy variable**. For simplicity I assume that noise always has zero mean since a non-zero mean is captured by the notion of bias. More precisely, if $\widehat{Y}$ is noisy then $\widehat{Y} = f(Y, \boldsymbol{X}) + X_0$ where X_0 is a random variable with mean zero, and samples $\widehat{y}$ from $\widehat{Y}$ are **noisy data**. Finally, I sometimes use the term **signal** to refer to the information gleaned about Y, separated from the bias and noise imposed by the observation process (e.g., biological signal). Typically some information about Y is lost in the transformation function $f(Y, \boldsymbol{X})$. Sources of measurement bias in avian monitoring include:

1. more difficult detectability based on season (Skirvin, 1981) and time of day (Robbins, 1981b; Skirvin, 1981; Rosenberg & Blancher, 2005; Hochachka *et al.*, 2009),
2. observer skill (Faanes & Bystrak, 1981; Bart & Schoultz, 1984; Sauer *et al.*, 1994; Kendall *et al.*, 1996),
3. misidentification of similar species (Faanes & Bystrak, 1981; Simons *et al.*, 2007),

4. weather (Robbins, 1981a), and

5. observer effort (Jobin *et al.*, 1996; Ferrer *et al.*, 2006).

A collection of samples may be biased even if each individual sample is free of measurement biases. **Sample selection bias** occurs when samples are non-randomly selected (Heckman, 1979). For example, the North American Breeding Bird Survey (Robbins *et al.*, 1986) only samples locations along road sides, leading to a road-side bias (Bart *et al.*, 1995; Hanowski & Niemi, 1995). To formalize the notion of sample bias, let $\boldsymbol{X}$ now be the independent variables associated with Y, the dependent variable of interest. Further, let ϕ be the probability density function for $\boldsymbol{X}$, and $\hat{\phi}$ be the density function used for selecting samples. The sampling density $\hat{\phi}$ is biased if there exists a value $\boldsymbol{x}$, within the domain of $\boldsymbol{X}$, such that $\hat{\phi}(\boldsymbol{x}) \neq \phi(\boldsymbol{x})$. Sample selection biases usually lead to biased statistics (estimators)—as defined in equation (2.3)—because statistics are computed from a random variable with a different distribution than the true distribution that determines θ.

2.4.2 Bias-Variance Decomposition

The machine learning community has applied the notions of bias and variance to describe and understand the performance characteristics of learning algorithms. Intuitively, a learning algorithm can be thought of as an estimator for the unknown function f that relates the predictor and response variables (i.e., $Y = f(\boldsymbol{X})$). Consider a single example $\boldsymbol{x}$. An **algorithm's bias** measures the difference between the algorithm's *main prediction* and the optimal prediction for $\boldsymbol{x}$ (the Bayes rate prediction). The main prediction will be explained below, but for now assume that it is the average prediction over all possible training sets of a fixed size. The **variance of an algorithm** is how much the algorithm's

prediction fluctuates over different possible training sets of a given size. *In summary, an algorithm's bias is its tendency to stray from the optimal prediction, and its variance is its instability when faced with different data samples.*

Interestingly, for some loss functions the error for a single example $\boldsymbol{x}$ can be decomposed into noise, bias, and variance components that are denoted $N(\boldsymbol{x})$, $B(\boldsymbol{x})$, and $V(\boldsymbol{x})$, respectively. This is called the **bias-variance decomposition** of loss. For example, the squared error for an example can be decomposed into the sum of noise, bias, and variance, all non-negative (Geman *et al.*, 1992). The noise is the intrinsic error or uncertainty for $\boldsymbol{x}$'s correct prediction, regardless of learning algorithm.

We adopt the notation and definitions from Domingos (2000) to formally express these quantities. Let $L(t, y) = (t - y)^2$ denote the squared loss of the prediction y for test example $\boldsymbol{x}$ which has the true value t. Further let $E_D[\cdot]$ be the expectation over the distribution of possible data sets of a fixed size; similarly, $E_t[\cdot]$ is the expectation over the distribution of possible true values for $\boldsymbol{x}$ (in a stochastic domain), and $E_{D,t}[\cdot]$ is over the joint distribution of D and t. Then expected squared loss for $\boldsymbol{x}$ can be decomposed as:

$$E_{D,t}[L(t,y)] = N(\boldsymbol{x}) + B(\boldsymbol{x}) + V(\boldsymbol{x}), \qquad N(\boldsymbol{x}) = E_t[L(t,y_*)]$$
$$B(\boldsymbol{x}) = L(y_*, y_m)$$
$$V(\boldsymbol{x}) = E_D[L(y_m, y)]$$

where y is the prediction from a model trained on data drawn from D, y_* is the **optimal prediction** that minimizes $E_t[L(t, y_*)]$, and y_m is the **main prediction** that minimizes $E_D[L(y, y_m)]$. For squared loss y_m is the mean prediction of the algorithm across possible training data sets. The expected bias and variance are computed by averaging over multiple test examples.

There are multiple proposals for the bias-variance decomposition of clas-

sification errors, also called zero-one loss (Kong & Dietterich, 1995; Kohavi & Wolpert, 1996; Tibshirani, 1996a; James & Hastie, 1997; Friedman, 1997; Breiman, 1998; Domingos, 2000). The formulation by Domingos is the most elegant in my opinion and is used to compute the bias and variance of zero-one loss in Chapter 6. Let $L_{0/1}(t, y)$ denote the zero-one loss function for classification:

$$L_{0/1}(t, y) = \begin{cases} 0 & \text{if } t = y, \\ 1 & \text{otherwise.} \end{cases}$$

The main prediction y_m for zero-one loss is the most frequently made prediction for the example (aka the mode). Let $P_D(y = y_*)$ be the probability over training sets in D that the learner predicts the optimal class for $\boldsymbol{x}$. Then the bias-variance decomposition for zero-one loss is:

$$E_{D,t}[L_{0/1}(t, y)] = c_1 N(\boldsymbol{x}) + B(\boldsymbol{x}) + c_2 V(\boldsymbol{x}) \tag{2.6}$$

where

$$c_1 = P_D(y = y_*) - P_D(y \neq y_*) P_t(y = t | y_* \neq t) \tag{2.7}$$

$$c_2 = \begin{cases} 1 & \text{if } y_m = y_* \\ -P_D(y = y_* | y \neq y_m) & \text{otherwise.} \end{cases} \tag{2.8}$$

The definitions of $N(\boldsymbol{x})$, $B(\boldsymbol{x})$, and $V(\boldsymbol{x})$ are the same as for squared loss. The expected noise and *net variance* for a test set become $E_{\boldsymbol{x}}[c_1 N(\boldsymbol{x})]$ and $E_{\boldsymbol{x}}[c_2 V(\boldsymbol{x})]$, respectively.

Under this formulation, noise and variance are sometimes beneficial and can decrease the expected classification error. When c_1 is negative, noise is beneficial; when c_2 is negative, variance is beneficial. Consider first the constant c_1. The noise term $N(\boldsymbol{x})$ is the loss incurred by the optimal prediction when the

function $f(\boldsymbol{x})$ is stochastic, and there is inherent uncertainty or ambiguity about what the correct classification for $\boldsymbol{x}$ is. When the $y = y_*$, the learner agrees with the optimal prediction and any noise increases error. If, however, the learner disagrees with the optimal prediction *and* the optimal prediction is wrong for $\boldsymbol{x}$, then there is a chance that the learner makes the correct prediction $(y = t)$. This lowers the expected error that is attributable to noise.

Now consider the constant c_2 and the effects of learner variance. If the learner's main prediction is good (i.e., agrees with the optimal prediction), then variance increases errors: the learner produces a model that predicts the main prediction less often. On the other hand, deviating from the main prediction is good if the main prediction disagrees with the optimal prediction; there is a chance that a particular model's prediction agrees with the optimal prediction $(y = y_*)$. In this way, variance can reduce the expected error by counteracting bias-related errors.

Bias and variance for an algorithm can be estimated on real data sets. I follow the same sampling framework used by Bauer and Kohavi (1999), since Bouckaert (2008) shows that bootstrap sampling results in less reliable bias-variance estimates. The train and validation sets are pooled to create D. Twenty samples of size $|D|/2$ are drawn from D without replacement. Each sample is used to train a model that makes predictions on the test set. This empirical distribution of the algorithm's predictions is used to compute expected bias and (net) variance according to the above equations.

In practice, one cannot know y_* for real data so I follow previous authors (Bauer & Kohavi, 1999; Domingos, 2000) in using $y_* = t$. As a result, the bias and noise cannot be separated and are combined in one term for our estimates.

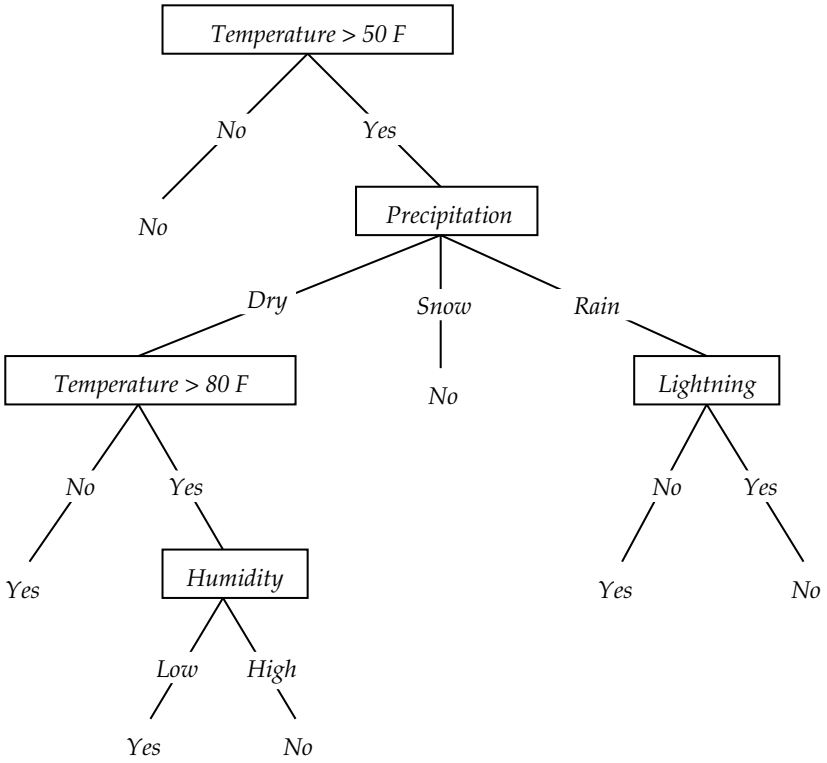

Figure 2.2: A decision tree for the concept PLAYULTIMATEFRISBEE. This tree classifies days based on their suitability for playing ultimate frisbee. An example (i.e., a day) is classified by sorting it through the tree into the appropriate leaf node and returning the leaf's classification (*Yes* or *No* in this tree).

2.5 Decision Tree Review

A basic understanding of decision trees is needed to understand Chapter 5. This section briefly reviews decision trees, how they are learned from data, and their strengths and weaknesses. The section ends with a brief synopsis of the different kinds of decision trees used in the dissertation.

Figure 2.2 depicts an example decision tree for classifying days as good or bad for playing ultimate frisbee.[3] Starting at the root of the tree (top of the figure), an example is classified by answering a series of questions (e.g., is the *Temperature > 50 F?)*. One question is asked at each internal node in the tree (the

[3]For a similar tree for playing tennis, see Mitchell (1997, Figure 3.1).

boxes in the diagram). Based on the answer to a question (written on the lines in the diagram), the example is passed down a branch of the tree to the next node. Each leaf node contains a classification that is applied to all examples reaching that leaf.

The statistics and machine learning communities independently developed algorithms to construct decision trees from data (Breiman *et al.*, 1984; Quinlan, 1993), but the core of the algorithms are the same. The tree learning algorithm proceeds in two phases: a *growing phase* that constructs a tree that can classify the training data, and a *pruning phase* that removes spurious tree branches (as determined by the tree's accuracy on validation data).

In the growing phase the learning algorithm starts by choosing a predictor variable to test; this test will become the root of the tree, and is used to partition the training data into two or more subsets. To find a good test, the algorithm searches through all the available predictors and scores how well each one partitions the data. Ideally, the resulting subsets will each contain a single class of example. In other words, the algorithm should prefer tests that reduce the impurity of the training sample by sorting the examples according to their classification labels. The most well-known scoring criterion in the machine learning community is information gain (Quinlan, 1986; Mitchell, 1997), a statistic that measures the reduction in entropy from testing a predictor.[4] Let K be the number of possible classes and S be a set of training examples. Then the set's **entropy** is defined as:

$$\text{Entropy}(S) = \sum_{i=1}^{K} -p_i \log_2 p_i \tag{2.9}$$

where p_i is the proportion of examples in S that belong to class i. Let A be a

[4]There are many other possible scoring functions. Surprisingly, the choice of scoring function does not have much impact on the accuracy of the trees (Breiman *et al.*, 1984; Mingers, 1989b; Buntine & Niblett, 1992).

predictor to score, and let $Values(A)$ be the set of values that A can take on.[5] Then the **information gain** from testing A is:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \tag{2.10}$$

where S_v is the subset of S for which predictor A has value v. The predictor with the highest information gain is installed as the root of the tree. The training samples are partitioned according to the installed test, and subtrees are recursively grown using the same process to classify the subsets of the training data. Tree growing stops and installs a leaf when either a) all examples in a subset share the same classification, or b) no test can be found to further categorize the examples in a subset. The classification associated with a leaf is the most frequent class in the training data that reaches the leaf.

If the training data contains noise, full-sized trees will overfit because spurious splits will be made to memorize the noise. The pruning phase of the learning algorithm solves this by classifying examples in the validation data and testing if performance can be improved by replacing various subtrees with leaves. There is some dispute about whether the choice of specific pruning algorithm significantly affects the accuracy of the final models (Mingers, 1989a; Esposito *et al.*, 1997), but studies agree that pruning generally improves performance. Although pruning is an important step in learning an accurate single decision tree model, it is much less important when learning an ensemble of decision trees (see Section 2.6).

Decision trees offer many benefits to analysts. First, decision trees gracefully handle discrete and continuous predictors. Second, continuous variables do not need to be normalized or scaled because the goodness of a split does not de-

[5]Continuous predictors are usually handled by sorting the values and finding a threshold to divide the values into high and low categories that can be scored using information gain. All possible thresholds are tried and scored.

pend on the variance of the predictor. Third, decision tree learning is efficient and scales well to large datasets. Fourth, a tree model can be studied and understood, provided the tree is not too large. Fifth, decision trees can be learned from noisy data and from data with missing values. In summary, decision trees are easy to apply to problems and can robustly learn in the face of noise and missing data.

The main drawback to decision trees is that the learning algorithm has high variance. Small changes in the training set can produce drastically different decision trees. Overfitting the training data is a serious problem that pruning only partially solves. As a result, decision trees tend to be less accurate than more recent machine learning models like artificial neural networks and support vector machines (Caruana & Niculescu-Mizil, 2006). However, decision tree learning also has low bias: with sufficient data a decision tree can be grown to approximate any classification function (Breiman *et al.*, 1984). The combination of high variance and low bias makes decision trees especially useful for ensemble learning, as will be discussed in Section 2.6.

A second well-known shortcoming is that decision trees can run out of data, particularly when trying to approximate functions that involve polynomial combinations of predictors (including linear combinations). Decision tree learning recursively divides the training data into smaller subsets. When only a single example remains after several tests, tree growing stops and the current tree node becomes a leaf because the subset cannot be further split. If a tree growing algorithm stops because of lack of data, the resulting tree may not be complex enough to represent the relationship between the predictors and the response variable.

Multiple decision tree variants appear in this dissertation. The differences

between tree types are not very important for this dissertation, but I summarize each type here for completeness. ID3 (Quinlan, 1986) trees are grown to full size using information gain as the splitting criterion and are not pruned. C4.5 (Quinlan, 1993) trees are the successor to ID3 trees. The main differences are the addition of tree pruning and using gain ratio as the splitting criterion. Gain ratio is a normalized form of information gain that controls for the arity of discrete predictors. CART trees (Breiman *et al.*, 1984) are similar, but use a more aggressive pruning rule that also considers the complexity of the tree. The pruned trees tend to be smaller than C4.5 trees. Three kinds of Bayesian trees (Buntine, 1992) are used: MML, SMML, and BAYES. These tree learning algorithms differ from the others in two ways. First, they place a prior on the tree structure that is combined with the training data to determine the posterior probability of possible tree structures. The posterior probabilities determine both which predictors are tested and when to stop growing a tree (sometimes called pre-pruning). Second, Bayesian trees are not pruned. Instead, predictions are smoothed by getting a prediction from the leaf and each of its ancestors on the path to the root; these fine- to coarse-grained predictions are combined in a weighted average. Minimum message length trees (MML trees) use Wallace's MML tree prior (Wallace & Patrick, 1993) and are large trees. Strict MML trees (SMML) use Wallace's full prior which prevents subtrees from branching too much (all but one branch from a test should lead to leaves). A BAYES tree builds multiple option subtrees at each interior node (rather than building the single tree with the highest posterior) and uses Bayesian averaging to combine the predictions of the option subtrees. See Buntine (1992) for full details.

Table 2.1: Cartoon demonstration of how an ensemble outperforms single models. Top of table shows the true value of the response variable for five examples. Each table row contains predictions from a single model; mistakes are italicized. The bottom row is an ensemble that makes predictions by voting models 1–5. Because the individual models make different mistakes, the ensemble can achieve perfect accuracy despite the fact that any single model is only 60% accurate.

Truth	**1**	**0**	**1**	**1**	**0**	**Accuracy**
Model 1	1	0	*0*	1	*1*	60%
Model 2	*0*	*1*	1	1	0	60%
Model 3	*0*	0	1	*0*	0	60%
Model 4	1	*1*	1	1	*1*	60%
Model 5	1	0	*0*	*0*	0	60%
Vote 1–5	1	0	1	1	0	100%

2.6 Ensemble Learning Review

Ensemble machine learning methods combine multiple base models (e.g., decision trees) to create a committee of experts (called an **ensemble**) whose performance is (hopefully) better than any of the constituent base models. Table 2.1 shows how an ensemble can outperform any single base model. "A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse" (Dietterich, 2000a, p. 1). Ensemble learning is more robust and reliable than non-ensemble learning because no single model needs to correctly predict all examples.

Bagging (Breiman, 1996) is a simple yet powerful ensemble method that is extensively used in this dissertation. **Bagging** is a meta-learning algorithm that repeatedly creates sub-samples of the training data and trains a model (e.g. decision tree) on each sample. Sampling is usually done with replacement. To make predictions, the bagged model averages the predictions of the constituent models (if models predict probabilities) or returns the plurality consensus vote

(if models predict classification labels). Bagging frequently improves the performance of a learning algorithm, and rarely hurts it. Bauer and Kohavi (1999) showed empirically that the main benefit from bagging is a reduction in the variance of the underlying models. Consequently, ensembles of bagged decision trees perform slightly better with unpruned decision trees (Bauer & Kohavi, 1999).

Bagging is a practical learning algorithm that is both simple and safe to run. Its performance is robust to noisy data, it is trivial to implement, it is trivial to parallelize, and there are no critical hyper-parameters to tune. Boosting (Freund & Schapire, 1996) and random forests (Breiman, 2001a) are two competing, more sophisticated ensemble algorithms. Both of these alternatives slightly outperform bagging on average in empirical comparisons (Bauer & Kohavi, 1999; Caruana & Niculescu-Mizil, 2006) but are less convenient to run in practice. The boosting algorithm requires care when implementing (special attention must be paid to avoid numerical overflows (Bauer & Kohavi, 1999)), cannot be easily parallelized, and overfits on tasks with noisy data (Grove & Schuurmans, 1998; Bauer & Kohavi, 1999; Dietterich, 2000b). This last point is particularly important: on some problems boosted ensembles are significantly more accurate than bagged ensembles, but on a noisy problem they can be worse than a single base model. Because of the potential to overfit, boosting is frequently applied to models with restricted flexibility (e.g., decision trees grown to a maximum height of 2 or 3). A less severe but related problem is that a boosted ensemble's accuracy can start degrading if the algorithm is run for too many iterations (i.e., the ensemble contains too many base models). To avoid this overfitting, a user should truncate a boosted ensemble to the first m learned base models (with m chosen to maximize performance on held-out validation

data) or prune detrimental base models (Margineantu & Dietterich, 1997). Random forests are similar to bagging in their robustness, ease-of-use, and ease of parallelization; further, they perform at least as well as boosting (Breiman, 2001a; Caruana & Niculescu-Mizil, 2006; Caruana *et al.*, 2008) and train faster than bagged or boosted tree ensembles when there are a large number of predictor variables. Unfortunately, random forests can only generate ensembles of decision trees, and as of 2010 it is still hard to find robust software packages implementing random forests.

The most popular base learning algorithm for ensemble learning is decision tree induction. Partly this is driven by efficiency: tens or hundreds of base models will be trained, so the base learning algorithm should learn models fast. More importantly, ensemble learning solves the main weakness of decision trees: high variance.

2.7 Variable Selection Review

Variable selection tries to find a subset of the available predictors that are relevant / useful for the given prediction task. Four reasons are traditionally given to motivate variable selection (Guyon & Elisseeff, 2003a): better predictive performance; computational efficiency from working with fewer inputs; cost savings from having to measure fewer variables; and simpler, more intelligible models. Different types of variable selection exist to satisfy varying balances of these competing goals under a variety of data regimes.

Indeed, the literature on variable selection is extensive, spanning the fields of machine learning, pattern recognition, and statistics. In statistics the problem of automating variable selection dates back to the 1960's (Efroymson, 1960), if not earlier, and is most often studied in the context of finding the best regression

equation (Draper & Smith, 1981). In machine learning the problem is known as feature selection and has been heavily studied since the 1990's, resulting in the 1994 AAAI symposium on relevance (Greiner & Subramanian, 1994; Subramanian *et al.*, 1997), and NIPS workshops on feature selection in 2001 (Guyon & Elisseeff, 2003b) and 2003 (Guyon *et al.*, 2006). Summarizing this vast literature is beyond the scope of this dissertation. Instead, I focus below on the two classic variable selection techniques that are used in Chapter 6. Readers interested in more complete surveys of variable selection should refer to Blum and Langley (1997) and Guyon and Elisseeff (2003a). The machine learning community's most recent advances in variable selection can be found in Guyon & Elisseeff (2003b), Guyon *et al.* (2006), and Saeys *et al.* (2008b).

This dissertation employs **forward stepwise variable selection** (FSVS) (Efroymson, 1960; Caruana & Freitag, 1994; Kohavi & John, 1997) and **correlation-based variable filtering** (CVF). FSVS is preferred when getting the best performance from the smallest variable set possible is important — as long as it is computationally feasible. For large data sets with hundreds or thousands of variables, simple filter methods like CVF are affordable and often surprisingly competitive. In the NIPS 2003 Feature Selection Challenge, "[s]everal high ranking participants obtain[ed] good results using only filters, even simple correlation coefficients" (Guyon *et al.*, 2005, p. 6). The main drawback to univariate filters like CVF is that they estimate the value of a variable in isolation, ignoring a) possible interactions with other variables, and b) redundant information contained in variables ranked higher (already selected).

Starting from an empty selected set, **FSVS** measures the benefit of adding each individual variable to the selected set. The benefit is measured by training a model using only the selected variables (including the variable under con-

sideration). The most beneficial (or least harmful) variable is added to the selected set, and the process is repeated for all remaining unselected variables. The search stops after a fixed number of steps, once performance has stopped improving, or after all variables have been selected. If feasible, the learning algorithm used in wrapper-based variable selection usually is the same algorithm to be used with the reduced variable set.

It is important for the search process to measure performance using data withheld during training to ensure good performance estimates. Additionally, the search process itself can potentially overfit this withheld data, so a third data set should be used to get an unbiased estimate of the selected subsets' performance (Reunanen, 2003).

CVF ranks the set of variables by their *individual* correlation with the class label. We use the magnitude of Pearson's correlation coefficient as the ranking criterion. The absolute correlation of predictor $\boldsymbol{x}_{\cdot j}$ with the label $\boldsymbol{y}$ is:

$$r_j = \frac{|\sum_i (x_{ij} - \overline{x}_{\cdot j})(y_i - \overline{y})|}{\sqrt{\sum_i (x_{ij} - \overline{x}_{\cdot j})^2 \sum_i (y_i - \overline{y})^2}}$$

where i indexes over examples and $\overline{x}_{\cdot j}$ and $\overline{y}$ are the respective means of $\boldsymbol{x}_{\cdot j}$ and $\boldsymbol{y}$.[6] Variables above a *cutoff* point are retained, while the others are discarded. Common strategies for selecting cutoff points include statistical tests of significance and cross-validated model performance at different ranks. We will be interested in the performance at varying rank-levels, so we do not need to choose a cutoff.

[6]The high dimensional data sets we use are all binary classification problems with binary and/or continuous variables, so Pearson's correlation coefficient is reasonable. Spearman's rank correlation would be a reasonable alternative for non-binary problems or nominal-valued variables.

2.8 Overview of Species Distribution Modeling

Species distribution modeling strives to link the population of a species to habitat, or more generally, environment features. Most often the goal is to predict a species' abundance (i.e., density) or probability of occurrence for a given place and time based on a description of the environment; occasionally other measures of population health and success, like survival rate and fecundity, are predicted (Stauffer, 2002; Van Horne, 1983). Section 1.2 explained why studying and understanding bird populations is both important and challenging.

Accurate models are an important component to understanding populations because their predictions fill in the gaps between data samples. The cost of directly measuring the abundance or frequency of a species in all locations (and times) of interest is prohibitively expensive for all but small study areas. A good model is a cost-efficient way to estimate abundance or frequency for large spatial extents. For example, Young et al. (2009) used distribution models to create abundance maps with a resolution of 1-km^2 for 115 avian species on the eastern slopes of the Andes in Peru and Bolivia; by overlaying the maps they identified areas important to species conservation.

Spatial scale is an important design point for distribution modeling (Guisan & Thuiller, 2005). There are two aspects to a model's scale: grain size and extent. The **grain size** for a model is its prediction resolution; grain size is determined by the size of the sampling area or the resolution of the predictors, whichever is larger. The **extent** is the area for which the model is valid (e.g., the continental United States), and is determined by the area from which training data are sampled. If the wrong extent is used for training the model, the resulting model may not meet the analyst's needs outside that extent. If the grain size is smaller or larger than the scale at which the species views the landscape, the training data

will appear noisy—likely hurting model performance. Coarse grain size also limits the resolution of maps generated from models. Unfortunately, choosing an appropriate grain size for modeling is non-trivial because it depends on the species being modeled, and thus far ecology does not have good theories about the inherent resolution of different species. The distribution models in later chapters are trained to predict over large extents (e.g., the continental United States) with coarse grain sizes (e.g., 15 km x 15 km in chapter 3 and 4 km x 4 km in chapter 5). Coarse grain sizes were used because of large sampling areas (in chapter 3) and limited resolution for climate predictors (in chapter 5).[7]

An inherent assumption underlying most distribution models, including the ones presented in subsequent chapters, is that the species' population is in **equilibrium** with its environment; i.e., its distribution is stable and will not change unless the environment changes (Guisan & Thuiller, 2005). Training data is sampled from a limited spatial and/or temporal snapshot, and a trained model will necessarily reflect the conditions of that snapshot. In practice, the equilibrium assumption is likely to hold for most species over short time frames but is almost certainly not true for any species over long time frames. The constant interaction between species results in a dynamic ecosystem that constantly (but usually slowly) adapts to new conditions.

Models of probability of occurrence are learned and studied in chapters 3 and 5. These models are often called presence-absence models because they are learned from **presence-absence data**: observations where a species is recorded

[7]Grain size and resolution for our data, and consequently our models, is actually more complicated. On the one hand, space is not gridded in advance so there is infinite resolution regarding the location of observations. On the other hand, the predictors associated with an observation are derived from raster data with fixed grids, the predictors have different resolutions, and the grids for different predictors are not aligned. The grain sizes reported here are the coarsest resolutions among the observations and predictors. Because our models are used to generate coarse maps, we have not taken the (immense) effort to reprocess and standardize the predictor resolutions. This *would be* an important pre-processing step for generating fine-grained maps of smaller spatial extents.

as present or not, without keeping track of how many individual of that species were detected. The rest of this section reviews presence-absence modeling in more detail and discusses how these models are evaluated.

2.8.1 Presence-Absence Modeling

Presence-absence models predict the occurrence of a species in an area but not the count (aka, abundance) of the species. Occurrence models are used for studying the range of a species and for ranking sites by likelihood of occurrence (often for conservation decision making).

There are two categories of occurrence models. Approaches in the first category rely on presence-only data; that is, information is only available about where the species was found, and no information is assumed for where they were not found. The current state-of-the-art method for presence-only modeling is MaxEnt (Elith *et al.*, 2006), a constrained and regularized density estimation approach based on the maximum entropy principle (Dudík *et al.*, 2007). Approaches in the second category use both presence and absence data; this includes our work in later chapters. Many approaches have been tried for presence-absence modeling (Scott *et al.*, 2002; Guisan & Thuiller, 2005; Hegel *et al.*, 2010); decision tree ensembles are the current state-of-the-art method (Moisen *et al.*, 2006; Stohlgren *et al.*, 2010; Fink *et al.*, In press). Generally, if absence information is available it is used. When species detection is imperfect (i.e., most of the time, for almost all species), absence information is noisy because a data collector (observer) may miss individuals that are present but hidden or quiet (see § 1.2 for possible reasons).

2.8.2 Evaluating Presence-Absence Models

As with all modeling, the appropriate performance measure for evaluating occurrence models depends on how the model will be used. The performances of the presence-absence models in this dissertation are measured using ACC, AUC, and RMS (defined in § 2.3) to evaluate diverse aspects of their accuracies. (Caruana & Niculescu-Mizil, 2004). These metrics respectively measure how often the model is correct, its ability to rank locations by likelihood of species occurrence, and the quality of the predicted probabilities.

ACC seems like an ideal fit for assessing occurrence models but in actuality it suffers from the need to choose a threshold and from its sensitivity to species prevalence (i.e., skewed class distributions). Fielding and Bell (1997) point out that changing the threshold for converting model outputs into discrete predictions of present or absent completely alters the performance of models. The default threshold of 0.5 (assuming raw outputs in the range [0,1]) will only be best if it is the best cutoff point for discriminating between positive and negative examples. Otherwise, better accuracy can be achieved using a different threshold. More troublesome, when one class is rare (e.g., a species occurs at a minority of locations in the modeled extent) high accuracy is achieved by always predicting the majority class (Manel *et al.*, 2001). Fielding and Bell review several other performance measures derived from a model's confusion matrix (see Table 2.2), and conclude that, like accuracy, all of them require choosing a suitable threshold and all but one are sensitive to class distribution. They recommend changing the threshold based on whether false positive or false negative errors are more serious for the model's intended application, and using measures that focus on the minority class instead of accuracy if species prevalence is low (e.g., precision and recall; see § 4.4 for definitions). Even better is to assign costs to

Table 2.2: Example confusion matrix from the house finch model from chapter 5. A **confusion matrix** tabulates how often a model's prediction (table rows) agrees or disagrees with the correct answer (table columns) on test data. Positive and negative correspond to present and absent for occurrence modeling.

	TRUE POSITIVE	TRUE NEGATIVE
predicted positive	16732	2876
predicted negative	2876	10030

each error type and measure the expected cost of the model's errors, but in practice appropriate costs are unknown and are difficult to assign (Fielding & Bell, 1997). Largely in response to Fielding and Bell's critique, the ecological community relies much less on accuracy as a performance measure than previously. It continues to be used, however, so we include it in our model assessments. We place less importance on ACC than on AUC and RMS though, due to the shortcomings above.

AUC measures the discrimination ability of a model without the requirement to choose a threshold. High AUC scores indicate that the model correctly ranks examples of presence above examples of absence. AUC is widely used to assess occurrence models because it is threshold-free and because the ability to rank locations by species occurrence rates often meets the needs of conservation planning to identify the best areas to conserve with a fixed budget (Fielding & Bell, 1997; Pearce & Ferrier, 2000; Pearce *et al.*, 2002). Some researchers have argued against using AUC (Manel *et al.*, 2001; Lobo *et al.*, 2008), however, pointing out that:

- AUC weighs omission (false negative) and commission (false positive) errors equally (Lobo *et al.*, 2008).
- Only the initial portion of the ROC graph, where the false positive rate is low, is interesting when species prevalence is low (i.e., a rarely occurring

species), and a model can have a good AUC because it performs well for unacceptably high false positive rates (Lobo *et al.*, 2008).[8] Note that *rare* in this context is in relation to the total extent being modeled. In particular, specialist species that occupy specific habitats may be easy to find in the correct habitat but only occur in a small percentage of observations across the entire study area.[9]

While these criticisms are true, we feel that AUC is nevertheless a very useful diagnostic. Both flaws are easily addressed when they are serious impediments for evaluating the utility of an occurrence model. Costs can be assigned for omission and commission errors, and the ROC curve can plot the benefit from true positives (normalized by the benefit from perfectly predicting the positive examples) vs. the cost of false positives (normalized by the cost of incorrectly predicting all the negative examples). The benefit for true positives may be defined implicitly as the benefit from avoiding the cost of false negatives. The second criticism is solved by computing partial AUC scores from the part of the graph with acceptable false positive rates (McClish, 1989; Dodd & Pepe, 2003).

Squared error (RMS) is not often used in ecological modeling for measuring the performance of presence-absence models, but it is the standard metric for regression modeling. We include it in our evaluations because it is a robust alternative to ACC that does not require choosing a threshold. Predictions further from the correct answer are more heavily penalized than predictions close to the correct answer. RMS also summarizes how well calibrated a model's pre-

[8]The converse problem arises in ecological niche modeling when the most important criteria is a low false negative rate; that is, only thresholds with high sensitivity / recall / true positive rate should be considered (Peterson *et al.*, 2008).

[9]For example, the gadwall (*Anas strepera*) is a common duck in lakes and ponds, yet it only occurs in 1.5% of eBird checklists during the summer (Munson *et al.*, 2009). If there are 10,000 examples in the test set, then a gadwall occurs in 150 of the examples; at a 5% false positive rate a model would have 492 false positives. Even if the model can detect all 150 positive examples at this threshold, only 23% of the model's positive predictions are actually correct.

dicted probabilities are, and good calibration is an important criterion for some applications of occurrence modeling. For example, a conservation organization might want reliable estimates of occurrence probabilities when deciding to purchase land for a future nature preserve so that cost-benefit tradeoffs can be computed.

The ecological modeling literature contains many other performance measures for evaluating occurrence models that are less popular than ACC and AUC; we end this section by mentioning two that appear to be growing in popularity and a third that measures accuracy with spatial weighting. First, Cohen's kappa coefficient (Cohen, 1960) measures the agreement between a model's predictions and the correct answer after correcting for agreements expected by chance (Fielding & Bell, 1997; Manel *et al.*, 2001; Hegel *et al.*, 2010). Second, a model's calibration can be explicitly measured and decomposed into refinement, bias, and spread components using techniques originally used for verifying weather forecasts (Pearce & Ferrier, 2000; Pearce *et al.*, 2002). Third, Fielding and Bell (1997) proposed spatially aware accuracy measures. The basic idea is that a false positive prediction should be penalized less if it is spatially close to a true positive. Distance is measured using a regular grid superimposed over the study's extent. This makes intuitive sense but requires that test data is available for all grid cells—a steep requirement for continent-scale models.

2.8.3 A Note on Independent Test Data

The occurrence of a species is spatially correlated. For example, if eastern bluebirds (*Sialia sialis*) occur at location A, they are likely to also occur at locations close to A.

The spatial correlation of species distribution has important consequences

for evaluating models. Specifically, care must be taken to ensure that test data are independent of the training data. At a minimum, training and test data should not be sampled from the same locations (e.g., on different days). Our initial work with modeling species occurrence (reported in chapter 5) did not take this into account; instead, we randomly partitioned the available data into training and testing subsets. Only later did we reevaluate the models using test data from independent locations and discover that the original performance estimates were overly optimistic. (Fortunately the conclusions in chapter 5 are still valid because the comparisons of importance measures do not depend on the estimates of model accuracy.)

But even data sampled from different locations are not guaranteed to be independent. Bird feeders at adjacent houses, and even in the same neighborhood, are likely to be visited by the same bird species; knowing which species visit a yard's feeders is incredibly helpful for predicting avian visitors to neighboring yards—*without knowing anything about the relationship of these species to the environment.* Unfortunately, how far apart locations need to be to ensure independence is an open question. The experiments in chapter 3 take a conservative approach and superimpose a checkerboard grid over the continental United States to partition the training and test data. Squares on the checkerboard measure 150 km to a side. Examples in white squares are training data, and examples in black squares are test data. This greatly reduces the chances of training and testing data being close enough to be spatially correlated. To our knowledge, this evaluation methodology is novel for ecological modeling.

CHAPTER 3

QUANTIFYING RELATIVE DATA QUALITY *

> *When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the state of* science.
>
> — Lord Kelvin, 1883 (Knowles, 1999)

Quality data are a crucial component to learning a good predictive model. Verifying the quality of large-scale data poses a challenge for analysts, particularly when the data are collected by networks of volunteers. In ecology, the pervasive access to the Internet has led to the development of several extensive monitoring projects that engage massive networks of volunteers who provide observations following relatively unstructured protocols. However, the value of these data is largely unknown.

This chapter describes a novel cross-data validation method for measuring the value of data from one source (e.g., a citizen science project) relative to a second, benchmark data source. The method fits a model to the data of interest and validates the model using benchmark data, allowing us to isolate the training data's information content from its biases. We also define a data efficiency ratio to quantify the relative efficiency of the data sources.

We apply our cross-data validation method to quantify the value of data collected in eBird—a western hemisphere, year-round citizen science bird checklist project—relative to data from the highly standardized North American Breed-

*Large portions of this chapter will be published as:
Munson, M.A., Caruana, R., Fink, D., Hochachka, W.M., Iliff, M., Rosenberg, K.V., Sheldon, D., Sullivan, B.L., Wood, C., & Kelling, S. (In press). A method for measuring the relative information content of data from different monitoring protocols. *Methods in Ecology and Evolution*. Accepted for publication April 2010.

ing Bird Survey (BBS). The results show that eBird data contain information similar in quality to that in BBS data, while the information per BBS datum is higher.

In ecology these methods can be more generally used to evaluate the suitability of sources of data for addressing specific questions for taxa of interest.

3.1 Introduction

Species monitoring is used for assessing conservation status, ascertaining and predicting the effects of habitat change, establishing management and conservation priorities, and determining how management efforts are meeting objectives (U.S. NABCI Monitoring Subcommittee, 2007). Effective monitoring requires that collected data are representative of the region of interest (e.g., Sauer, 2000), that data collection methods ensure a relatively high probability of detecting the target species (e.g., Conway & Timmermans, 2005), and that data can be adjusted for imperfect detection of organisms (e.g., MacKenzie, 2006). Designing ideal protocols to meet these goals would likely require protocols tailored to each individual species, which is likely impractical. Thus, species monitoring is typically accomplished through more generic data-collection schemes. While data from such programs have their weaknesses (Nichols & Williams, 2006), they are useful resources for finding answers to unanticipated questions for which no data exist from a "well-designed" monitoring scheme. For example, much of our knowledge of climate change and its effects on distribution (e.g., Thomas & Lennon, 1999; Hickling *et al.*, 2005) and demography (e.g., Jiguet *et al.*, 2006) is based on data from generic monitoring schemes. Further, it is not always the case that a researcher has sufficient knowledge of the important processes to design studies and create protocols and statistical models that

are appropriate to the biological objectives (Fitzpatrick *et al.*, 2009). The prior knowledge needed to appropriately design targeted studies must come from somewhere, and generic monitoring schemes can provide the needed insights to effectively design targeted studies (Hochachka *et al.*, 2007).

One relatively unexplored source of generic, surveillance monitoring data is from low-structure, high-volume checklist programs, whose data value has not been thoroughly investigated, although focused demonstrations of information content in the data exist (e.g., Hochachka & Dhondt, 2000; Koenig, 2001; Kéry *et al.*, In press). For birds, numerous checklist-based monitoring projects exist in the Americas (Sullivan *et al.*, 2009), and elsewhere (Schmid *et al.*, 2001; Baillie *et al.*, 2006; Harrison *et al.*, 2008). Often these projects engage large numbers of volunteers, making the cost per datum trivial (Sullivan *et al.*, 2009). The utility of these data depends not only on how they were collected but also on the species, performance measures, and questions of interest; data quality cannot be measured in the abstract. *What is needed are general methods for quantifying the effectiveness of different monitoring schemes for answering particular questions about species of interest.* Such methods would identify the most useful data source for meeting an analyst's needs. However, comparing monitoring schemes is complicated by the fact that each monitoring program has its own sets of biases and sources of variance. For example, the North American Breeding Bird Survey (BBS; Robbins *et al.* 1986) is a highly standardized survey conducted along roadsides, which influences how frequently different species are detected (Bart *et al.*, 1995; Hanowski & Niemi, 1995, e.g.,). In contrast, the eBird project (Sullivan *et al.*, 2009) allows volunteers to conduct surveys wherever they want to watch birds, resulting in much denser data near major population centers (Fig. 3.1).

In this chapter we develop a **cross-data validation** method for quantifying

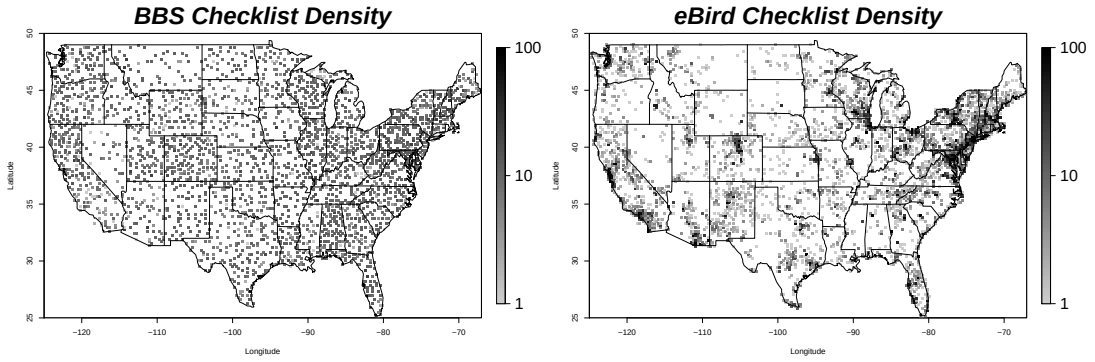

Figure 3.1: BBS data are collected using a stratified design to ensure roughly uniform spatial coverage; eBird data are collected wherever birders choose to go, and is denser where people live. Each pixel shows the number of submitted checklists within a 20 km by 20 km square; white indicates no checklists were submitted during May–July time period (2003–2008).

the predictive power of a data source relative to a benchmark data source. For concreteness, we present the method in the context of monitoring species status and distribution, but the method is general and can be applied to other predictive tasks (e.g., monitoring population trends) and domains (e.g., natural language processing). Our method compares the predictive power of two models (species distribution models in our example). The first model is learned using the data source of interest (called **candidate** data in the rest of the paper), while the second is learned using the benchmark data. We define a *data efficiency ratio* to quantify how many candidate data samples are needed to equal the information in a single benchmark sample.

To demonstrate the method, we quantify the value of eBird breeding season data relative to BBS data, for 75 regularly occurring North American bird species. eBird is a general purpose data source that collects bird observations year-round from any location in the western hemisphere. This generality is a potential boon for studying bird populations at times and places not covered by established monitoring programs. In particular, established large-scale mon-

itoring programs are limited to the breeding season, and relatively few data are available about bird populations during migration and winter. However, the reliability of eBird data, which is collected via a low-structured protocol, needs to be verified before scientists can use the data with confidence. Our case study compares the general purpose eBird data to the more specialized BBS data because the BBS is an established and widely used data source, with a highly structured protocol, whose reliability has been extensively studied. We mitigate the differences between the two data sources by choosing a subset of eBird data that approximates the characteristics of BBS data. The results show that eBird data contain breeding season distribution information comparable in quality to that in BBS data. More generally, our case study demonstrates that data from a low-structure data-collection protocol contain useful information; with enough such data, the information can be similar in quality to that collected using a more intensive, structured data scheme.

3.2 Methods

3.2.1 Cross-data Validation Framework

Directly comparing the information in the candidate data set to that in the benchmark data set is hard because projects use different sampling designs and protocols, and draw samples from different points in space and time. To overcome this, our approach is to fit a pair of models (using the respective data sets) that can predict the independent variable (e.g., species occurrence) as a function of predictor variables (e.g., habitat, climate). The predictive models generalize from the available samples to any point in space/time, and summarize the information available from the two data sets. We can directly compare the models'

predictions for a common set of independent test points from the benchmark data source.

Validating the candidate model on external, independent data gives us a way to objectively measure *data* quality. If we instead validated the candidate model using held-out candidate data, we could verify whether models could be fit to predict the combined biological and observational processes generating the candidate data. This would let us measure how well the *model* represents the data, but would not tell us anything about the value of the data themselves: the candidate data could be dominated by biases or noise that hide the underlying biological signal.

Because the models serve as data set proxies, they need to faithfully represent the information in the data. Parametric models can be used if the phenomenon being described is sufficiently well understood. However, for many species, the causal factors that relate to their occurrence are not well understood, and machine learning techniques that automatically construct accurate non-parametric models from data (Hochachka *et al.*, 2007; Kelling *et al.*, 2009) are more appropriate.

Two types of comparisons can be made between data sources. First, how does the information from candidate data compare to the information from benchmark data? To answer this, we summarize the candidate model's predictive power on benchmark test data using appropriate performance metrics (e.g., accuracy, mean squared error). We compare the candidate model's performance against the performance of the benchmark model and a simple baseline. Since the benchmark model is fit using training data drawn from the same distribution as the test data, it should demonstrate the best possible performance for the test data. Comparisons with the simple baseline indicate if either model

has learned anything meaningful. For this study the baseline was to always predict the frequency of the species in the benchmark data. For example, western meadowlarks (*Sturnella neglecta*) were recorded in 23.6% of BBS surveys, so the baseline predicted 0.236 probability of occurrence for all test surveys.

Second, how many candidate data samples are needed to achieve the same performance as the benchmark data? To estimate data efficiency, we vary the volume of training data used for constructing the models and observe how performance depends on training set size. By fitting the observed trends, one can estimate how many candidate data would be needed to equal the performance of the benchmark model. The **data efficiency ratio** is the ratio of candidate to benchmark data at the best benchmark performance level; the ratio quantifies the relative efficiency of the candidate data source. (See Fig. 3.4a for example.) In our study we found that performance improved roughly linearly with the log of data size. Accordingly, we fit the log-linear model:

$$y = a \log_{10} x + b$$

where x is the data size and y is the performance loss.[1]

The interpretation of a data efficiency ratio depends on the relative improvement rates of the two models. If the observed trends are parallel, then the data efficiency ratio is a threshold: as long as the ratio of candidate to benchmark data collected reaches the threshold, a candidate model can perform as well as a benchmark model. If the trends converge (e.g., Fig. 3.4b), the candidate model is improving faster than the benchmark model and is projected to perform as well as the benchmark model in the future (provided at least as many candidate as benchmark data are collected). If the trends diverge (e.g., Fig. 3.4c), the

[1]If performances for large data volumes are close to perfect, a log-quadratic model may be a better fit for the trend; intuitively, improving performances are more likely to asymptote to perfect prediction than to actually achieve prefect prediction with finite training data.

candidate model is falling behind and matching the benchmark's performance requires ever more data. The candidate model is unlikely to ever perform as well as the benchmark model unless one trend changes direction.

When available, experts can compare the models qualitatively and identify any differences that are not captured by the performance statistics. Experts possess a broader, more holistic view of species distribution that comes from synthesizing multiple sources, including their own field experience. Benchmark as well as candidate models can be verified by experts. We use distribution maps generated from each model to visualize and diagnose the information captured by complicated models.

3.2.2 Calibration

Consistent differences in the biases of the candidate and benchmark data sets can exaggerate performance differences. For example, the candidate model may consistently predict lower occurrence probabilities. Correcting such simple differences can significantly improve the candidate model's performance and give a clearer picture of how similar the two models are.

In preliminary analyses of data from five species, we found that eBird occurrence models were poorly calibrated when predicting BBS test data. To fix this problem, we post-calibrated eBird models after they were fit using Platt's scaling (Platt, 2000), which frequently produces good results (Niculescu-Mizil & Caruana, 2005). This calibration involves fitting the sigmoid function:

$$f(x) = \frac{1}{1 + \exp\{-(w_0 + w_1 x)\}} \tag{3.1}$$

where x is a model's raw prediction. The sigmoid function transforms a raw prediction (x) into a calibrated prediction ($f(x)$). A small amount of benchmark

data (i.e., set aside BBS data) was used to fit equation (3.1) for each model. Calibration noticeably improved the accuracy and mean squared error of candidate models for the preliminary results. Much of the benefit comes from adjusting the present/absent threshold (for accuracy) and correcting the mean probability (for mean squared error). No calibration was used for the benchmark models because preliminary results showed calibration hurt benchmark performance.

3.2.3 Data Collection and Processing

Our case study measured the quality of eBird breeding season data, relative to benchmark data from the North American Breeding Bird Survey (BBS). BBS data were chosen because they were collected using a stratified design with controls that minimize the impact of weather, time of day, and observer-based variance. The BBS data are the highest quality large-scale data set we know of for North America, and have relatively uniform coverage (Fig. 3.1). To compare data sets as fairly as possible, we selected subsets of each to make the data as similar as possible (details below).

We treated observations in both data sets as transect surveys where the observers travel along a route and record which bird species are detected. We did this because latitude and longitude coordinates were only available for the first stop of each BBS route, and location coordinates were needed to associate predictor variables (e.g., habitat) with each survey. Our analyses used the subset of surveys collected in the contiguous United States because most predictors were not available elsewhere. In the rest of the paper, a **survey** is a sampling event that records an entire route's observations for a specific date and time. Data sets may contain multiple surveys of the same route (e.g., from different years).

We converted these data sets from count data to presence/absence data; i.e.,

non-zero counts became *present*, and all others became *absent*. Because we used the predictive performance of the models as a proxy for the information in a data set, it was important that the learned models be as accurate as possible. Our experience is that modeling relative abundance is more challenging (and less well understood) than modeling occurrence. A poorly fit model can lead to the false conclusion that a data set contains little biological information; by studying occurrence modeling, we reduce this risk.

Occurrence models were fit for the 75 species most frequently recorded on BBS surveys (listed in Table 3.2). We assumed that regularly occurring species would have enough presence observations to allow accurate modeling, and be easy to detect (implying that benchmark data measures biology mostly unobscured by detection processes).

eBird Data

eBird (Sullivan *et al.*, 2009) does not enforce a stratified sampling design, resulting in uneven spatial sampling (Fig. 3.1). Importantly, volunteers answer questions about how each survey was conducted:

- Where were the data collected?
- When were the data collected? (Date and start time.)
- How many people were in the birding party?
- What kind of survey was conducted? Surveys can be casual counts (made while doing something else), stationary point counts, transect counts, or area counts (all birds within an area). For the last three types volunteers answer additional questions about survey duration and the distance or area covered (as appropriate). This information about effort is important for explaining some of the variance in the observations.

- Is the survey complete? On a complete checklist, the submitter reports all species they were able to identify.

eBird data contain both sample selection and measurement biases (Sullivan *et al.*, 2009).[2] In addition to the higher sampling density near population centers, eBird survey locations are also biased towards areas birders believe will have greater diversity and abundance, or species unusual for the region. This bias is especially acute at well-known locations for interesting birding that include many parks, refuges, and preserves managed specifically for birds. As with most bird monitoring schemes, species detectability is a problem and cryptic species are likely underreported. Despite a multi-tiered review process that catches many mistakes, misidentification remains a potential bias since eBird attracts participants with varying skill levels. Finally, one source of noise for our analyses is the non-standardized reporting of location for transect surveys; observers are encouraged to report the middle of the route as the location, but many report the beginning or end. This introduces extra noise into the spatial covariates associated with surveys.

We used the eBird Reference Dataset (Munson *et al.*, 2009), which excludes casual counts and incomplete surveys. By only using complete checklists we were able to infer which species were undetected (as opposed to certainly absent: "absence" information is noisier than presence information).

To make the eBird and BBS data similar, we selected transect surveys covering at most 8 km, that were collected from May through July, during the years 2003–2008 (2003 was the first year eBird collected at least as many transect surveys as the BBS (Fig. 3.2)). The number of observers was missing for 9% of surveys; we filled in these missing values with the mean number of observers

[2] See section 2.4 for definitions of biased and noisy data.

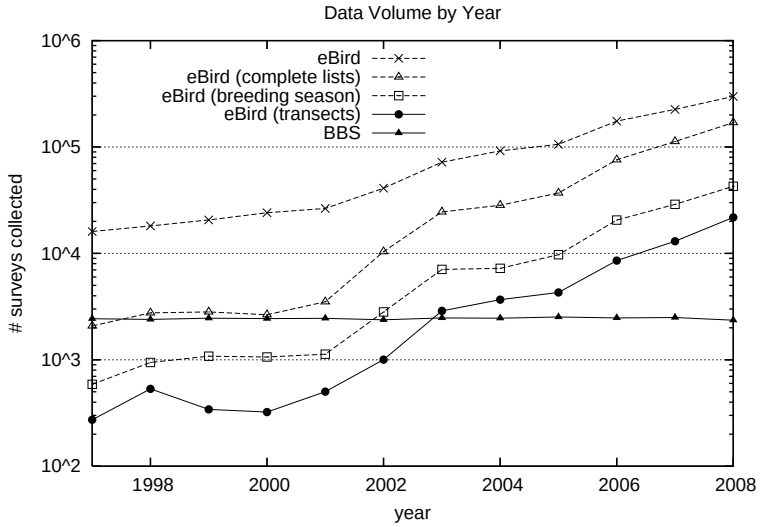

Figure 3.2: eBird data are much more abundant than BBS data. While BBS data per year are constant, eBird data per year are growing. The multiple eBird lines show data volume after successive data selection steps: complete, non-casual checklists only; breeding season only; transect surveys only. In 2008, eBird collected 8.7 times as many breeding season transect surveys as BBS.

(two).

BBS Data

The North American Breeding Bird Survey (BBS; Robbins *et al.* 1986) collects data about birds throughout much of road-accessible North America during the breeding season. Volunteers conduct road-side surveys along predefined routes that are distributed to ensure good spatial coverage. Each route consists of fifty stops spaced 0.8 km apart; at each stop the observer counts how many birds they detect within a 3 minute period. Routes are surveyed once per year at the height of the breeding season when birds' plumage and singing maximizes their detectability. Surveys start 30 minutes before sunrise since birds tend to be most active around sunrise.

BBS data contain several documented biases. First, the surveys sample road-side habitats, which may cause some species to be under-sampled (Bart

et al., 1995; Hanowski & Niemi, 1995). Second, observer skill with the protocol changes over time, both improving with experience (Kendall *et al.*, 1996; Sauer *et al.*, 1994) and worsening with hearing loss (Faanes & Bystrak, 1981). Third, observers faced with multiple vocalizing birds typically undercount the number of species and individuals per species due to the short observation period (Bart & Schoultz, 1984; Robbins *et al.*, 1986). The short duration also lowers the probability of detecting cryptic species (Jobin *et al.*, 1996).

Our analyses used BBS surveys from 2003–2008 collected in the contiguous United States. Only surveys of acceptable quality were used. (Each BBS survey is annotated with a variable called RunType that indicates if the USGS considers the survey acceptable for analysis.) We aggregated the first ten stops in each route into an 8 km transect count associated with the route's starting location. We omitted the later stops because 1) the predictors may not describe stops farther from the starting location as accurately; and 2) later stops occur farther from sunrise, resulting in decreased detection probabilities for most bird species (Robbins, 1981b; Skirvin, 1981; Rosenberg & Blancher, 2005; Hochachka *et al.*, 2009).

The eBird models required effort predictors to predict occurrence on BBS data. We set survey distances to 8 km and durations to 30 minutes, for all BBS surveys.

Predictor Variables

Predictor variables for the occurrence models included covariates describing the local region around the survey route and how much effort the observer expended (Table 3.1). The timestamp and observer covariates came with the eBird and BBS count data; the other covariates came from external GIS databases, and

were linked to the surveys using their location and time (Munson *et al.*, 2009). We chose to use large extent habitat predictors to ensure that the extents included the full length of the survey transects. We encoded missing categorical predictor values as MV (to treat them as just another nominal value) and discarded the small number of surveys with missing continuous values to avoid decisions about imputing missing values (58 BBS surveys; 1,856 eBird surveys).

We did not discretize the spatial extent into a fixed grid; instead, spatial predictor values were defined from the neighborhood around each survey location. Consequently, two surveys will only have identical spatial predictor values if they are within 30m of each other (the finest resolution of predictors used). The coarsest spatial predictor is 15km by 15km.

3.2.4 Occurrence Models

We trained bagged decision tree models (Breiman, 1996) to predict species occurrence.[3] We chose bagged decision trees for three reasons. First, many of the 75 species are not sufficiently well understood to specify parametric models. Second, bagged decision trees are flexible and powerful enough to approximate any function (provided enough training data are available) (Breiman *et al.*, 1984), giving us confidence that a bagged tree model would be able to detect and use any information signals in the data. Third, decision trees, bagged trees, and boosted trees have all been successfully used for species distribution modeling (De'ath & Fabricius, 2000; Caruana *et al.*, 2006a; Elith *et al.*, 2006; Hochachka *et al.*, 2007). We used the IND decision tree package (Buntine & Caruana, 1991) to train 100 MML-style decision trees (§ 2.5) for every occurrence model. Given the covariate description of a survey, each model predicts the probability of oc-

[3]Bagging and decision trees are described in sections 2.6 and 2.5.

Table 3.1: Summary of predictor variables used in models.*

Category (# Predictors)	Description
Checklist timestamp (3)	The year, day (1–365), and time survey was started. Source: eBird and BBS.
Observer (3)	Duration of observation (in hours), distance traveled (km), and number of observers in birding party. Source: eBird and BBS.
BCR (1)	Bird conservation region (numeric identifier). Source: shape files from ESRI in 2004. Resolution: n/a (BCRs are large and stretch across multiple states).
Climate (10) †	Averages over 30 years. Total precipitation for month. Average, min, and max daily temperature for month. Mean, median, and extreme data ranges for a) last 32F day in spring, and b) first 32F day in autumn. Source: Climate Atlas of the US, v2 (1961–1990), from NOAA-NCDC. Resolution: 4km by 4km. Details: `http://www.ncdc.noaa.gov/oa/about/cdrom/climatls2/info/atlasad.html`
Elevation (1) †	Elevation in meters. Source: National Elevation Dataset from USGS. Resolution: 30m by 30m. Details: `http://www.usgsquads.com/elevationdata.htm#NED_Info`
Human population (1)	Population per square mile (2000 US census). Source: shape files from ESRI. Resolution: n/a (census block-groups are variable size). Details: `http://www.census.gov/geo/www/tiger/glossary.html`
Habitat (16) †	Measures percent of surrounding landscape for each of 16 land cover classes (e.g. open water, deciduous forest). Landscape is 15km x 15km box around location. Remote sensing data from 2001 National Land Cover Database. Source: MRLC. Resolution: originally 30m by 30m, aggregated into 15km by 15km. Details: `http://www.mrlc.gov/nlcd.php`

* See (Munson *et al.*, 2009) for processing details.

† Predictors from raster data use different grids (i.e., grids are not aligned).

currence for a particular species. Recall that covariates describe the neighborhood centered around a survey location; hence, predictions are made separately for each survey location, and not, as is sometimes done in species distribution modeling, for each cell in a fixed grid.

3.2.5 Model Validation

We validated all occurrence models by measuring their predictive power on independent data. The data were divided into train and test sets (for model fitting and validation, respectively) according to a checkerboard grid with 150 km-sided squares. For example, surveys located in white squares were training data, while surveys in black squares were test data. The same grid was used for dividing the BBS and eBird data to avoid any chance of spatial overlap between train and test sets. (See § 2.8.3 for details on why spatial overlap is undesirable.)

After partitioning, there were roughly 21,175 surveys in the eBird training set, and 6,460 in both the BBS training and test sets. For calibrating model predictions, 300 surveys were subsampled from the BBS training sets and set aside ($\approx 5\%$ of training data). For the analysis of performance as a function of data size, data was subsampled from the remaining training data. All subsampling was by route; i.e., when one survey from a route was added to a subsample, all other surveys from that route were also added. For example, eight surveys were collected from each BBS route (on average), so the calibration set contained data from roughly 38 routes. Sampling by route ensured independent locations for training and calibration sets and simulated the benefit from monitoring more locations.

Predictive power was measured using accuracy (ACC), root mean squared

error (RMS), and the area under the ROC curve (AUC).[4] Accuracy is the percentage of times that a model correctly predicts if the species was present / absent. Root mean squared error measures the average error of a model's predicted probabilities. AUC measures how well a model ranks sites from high to low occurrence. All performance measures were computed using PERF[5].

In addition, we generated occurrence maps from each model to visually compare the eBird and BBS models. Each map shows the predictions of a model made for 130,000 random locations selected using a spatially stratified design. The model made predictions based on the covariate description for each location, with fixed values for the effort covariates (1 observer surveying an 8 km transect over a 30 minute period). Multiple predictions were made for each location, varying as a function of date (7 June, 14 June, 21 June, 28 June, for all years 2003–2008), and averaged to create a map representing the breeding distribution. Each map pixel is approximately 15km by 15km. For pixels that contain multiple random survey locations, the color is determined by the average of the predictions from those locations; the few pixels containing zero predictions are white.

Each analysis was repeated 20 times using different train/test splits. Ten different checkerboard grids were generated by randomly shifting the corner of the board. Two runs were made from each board: one run using white squares for training data, and one run using black squares for training data. All reported numbers and maps are averages over the 20 repetitions. Error bars in graphs show one standard deviation.

[4] ACC, RMS, and AUC are defined in section 2.3.

[5] `http://www.cs.cornell.edu/~caruana/perf/`

3.3 Results

3.3.1 Quality of eBird Breeding Season Data

To visualize the results for all 75 species, we plotted BBS vs. eBird performance. For most species, the eBird models were nearly as accurate as the BBS models (Fig. 3.3, middle column).

Calibration significantly improved ACC and RMS for most species (Fig. 3.3, left vs. middle), although it did slightly hurt model performance for 15 species (see Table 3.2). Calibration-induced errors fell into two categories. 1) Occurrence increased from zero to 5%–15% outside the species' range, as a result of raising occurrence within the range to match higher BBS frequencies. 2) Occurrence decreased to nearly zero in the edges of the species' range, as a result of correcting occurrence probabilities in the core of the range.[6] A more complicated calibration method might be able to overcome these flaws.

Most species were observed in the minority of surveys; accordingly, the baseline sometimes achieved good ACC by always predicting not present and good RMS by always predicting low probability of occurrence. To rule out the possibility that eBird appeared similar to BBS because the BBS models were close to the baseline, we compared the performance gaps of the eBird and baseline models. A model's (or baseline's) **performance gap** is the difference between its performance and the benchmark (BBS) model's performance. For all metrics, eBird models had smaller gaps for more species than the baseline (Fig. 3.3, right column). Even for ACC, where the baseline performed best, eBird was no-

[6]Mathematically, the cross-entropy objective used for tuning calibration penalizes larger differences in probability more than smaller differences. If the uncalibrated eBird model predicted much lower occurrence values than the BBS model for the core of the range—due to lower detection probability from collecting data in the middle of the day, for example—calibration corrected the occurrence probabilities in the core while mostly ignoring everywhere else.

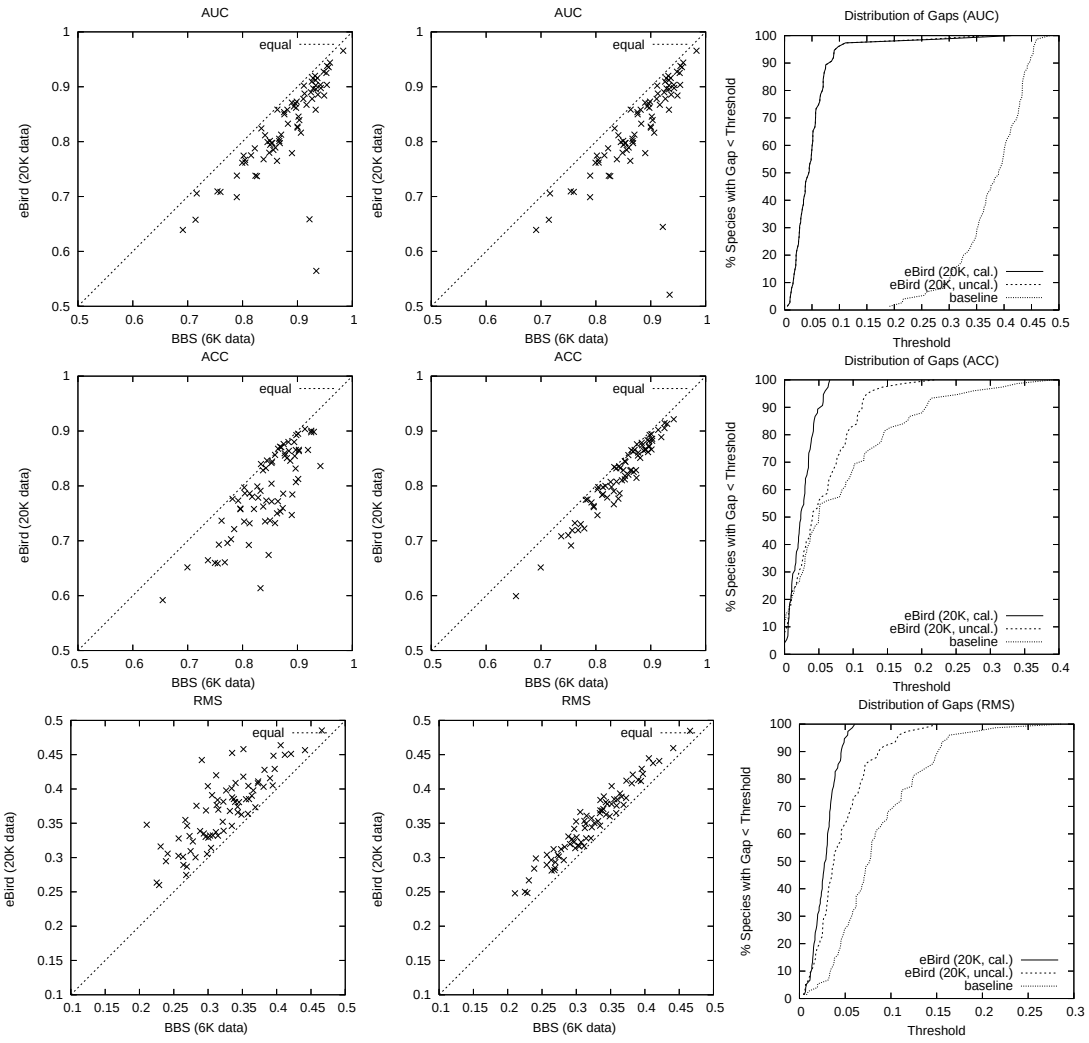

Figure 3.3: Performance of eBird models vs. BBS models. Scatter plots compare BBS performance to uncalibrated eBird performance (left column) and calibrated eBird performance (middle column), with one point per species. Points close to the diagonal line indicate similar BBS and eBird performance. The right column graphs show the cumulative distribution of **performance gaps**—the difference between BBS and eBird (or baseline) performance. The percentage of species with performance gaps less than various thresholds (x-axis) is shown for calibrated (*eBird (20K, cal.)*) and uncalibrated eBird models (*eBird (20K, uncal.)*), and for the *baseline* model. For example, calibrated eBird models had accuracy (ACC) at most 0.05 (5%) below the respective BBS models for 90% of species. The calibrated and uncalibrated lines are the same in the top right graph because Platt scaling does not change rankings.

ticeably better. Interestingly, without adjusting the threshold for accuracy (via calibration), eBird ACC was seemingly no better than baseline ACC for performance gaps of 0.05 or less; with the correct threshold, the models were clearly much better than the baseline.

3.3.2 eBird Data Efficiency

Data efficiency ratios greatly varied across both species and performance metrics (Fig. 3.4; Table 3.2). Overall, the eBird data were noisier than the BBS data due to non-uniform spatial sampling, lower detection probabilities, or varying survey lengths and durations. In general, ratios for AUC performance were the lowest, followed by ACC, and then RMS. Similarly, the performance trend lines for AUC loss were parallel or converging for 2/3 of the species studied—far more often than for ACC and RMS (Fig. 3.4d). Some diverging data efficiency ratios were very large or even infinite (Table 3.2). Infinite ratios occurred when the eBird model's performance worsened with increasing amounts of training data.

3.3.3 Expert Opinion of Maps

Overall, the BBS maps were slightly better than the eBird maps. For some species, the BBS and eBird maps were both very good and differed only in small details (e.g., Fig. 3.5). In other cases, both maps captured the species' range reasonably accurately but differed in the predicted occurrence due to differences in the data sources (e.g., Fig. 3.6). For some species, one or both of the maps contained major mistakes (e.g., Fig. 3.7).

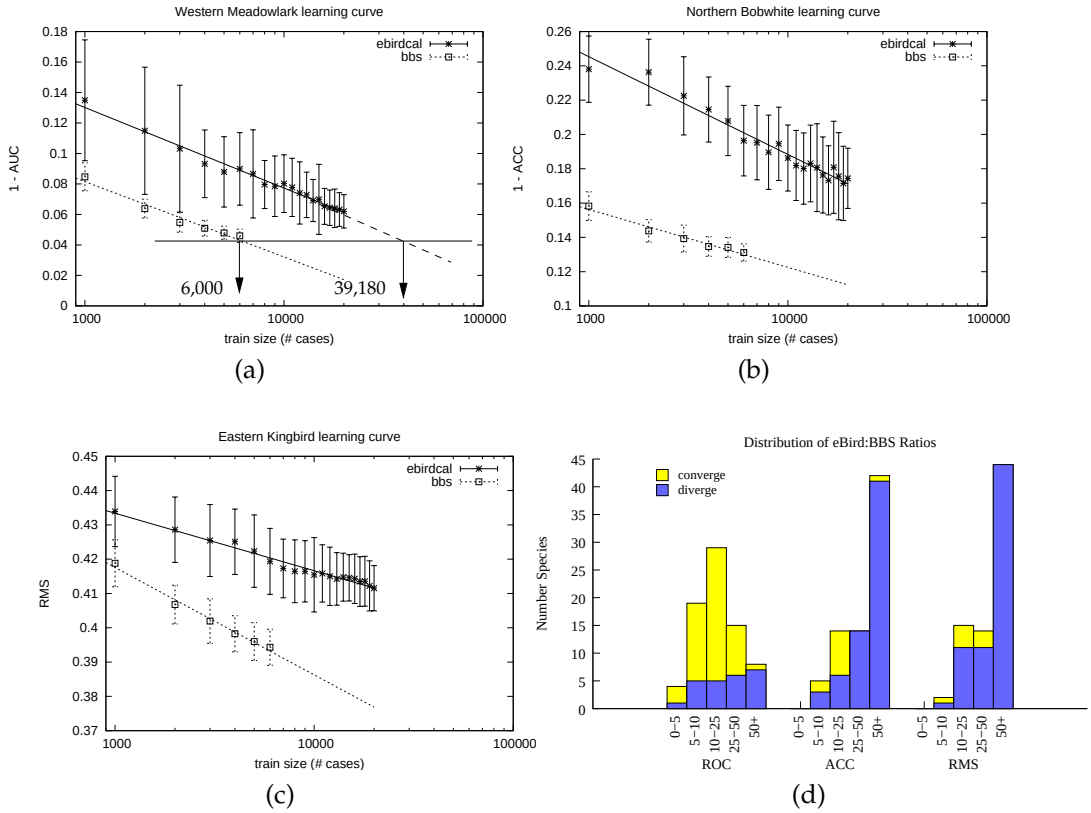

Figure 3.4: eBird's efficiency at collecting useful information, relative to BBS, depends heavily on the focal species and performance measure. (a) Example of estimating a **data efficiency ratio**. AUC loss for western meadowlark (*Sturnella neglecta*) models decreases as a function of training data size. Roughly 6.53 ($\frac{39{,}180}{6{,}000}$) times as many eBird data as BBS data are needed to match the best BBS model performance, based on fitting log-linear models to the performance trends. (b) The ACC trend lines for northern bobwhite (*Colinus virginianus*) are **converging**; i.e., the eBird model improves faster than the BBS model and should eventually catch up. (c) The RMS trend lines for eastern kingbird (*Tyrannus tyrannus*) are **diverging**; i.e., the BBS model improves faster than the eBird model. The eBird model will never match BBS model performance, unless one or both trends change direction in the future. (d) Stacked histograms of eBird:BBS data efficiency ratios. Each bar counts number of species with a data efficiency ratio in that range. If the trend lines used for estimating a ratio diverged, the ratio was categorized as diverging; otherwise it was categorized as converging.

Table 3.2: Data efficiency ratios for eBird compared to North American Breeding Bird Survey (BBS). Each number is the ratio of eBird to BBS data volume required for an eBird model to match best current BBS model performance (6K training data). Italicized ratios indicate eBird and BBS performance is diverging as training data increases; i.e. BBS performance improves faster than eBird performance as data grows. Infinite ratios (∞) indicate divergence and that eBird performance worsens as data grows.

SPECIES	AUC RATIO	ACC RATIO	RMS RATIO
American Crow (*Corvus brachyrhynchos*)	9.39	18.7	*27.2*
American Goldfinch (*Carduelis tristis*)	*52.2*	∞	*1.2e+04*
American Robin (*Turdus migratorius*)	14.2	*17.7*	*32*
Baltimore Oriole (*Icterus galbula*)	*6.52*	*248*	*22.1*
Barn Swallow (*Hirundo rustica*)	25.2	*633*	*55*
Black-capped Chickadee (*Poecile atricapillus*)	*12.8*	*65.3*	*55.2*
Blue-gray Gnatcatcher (*Polioptila caerulea*) *	34.1	*1.83e+07*	*1.59e+04*
Blue Grosbeak (*Passerina caerulea*)	19.3	*1.41e+03*	*537*
Blue Jay (*Cyanocitta cristata*)	*27.9*	*51.4*	*46.6*
Brewer's Blackbird (*Euphagus cyanocephalus*)	∞	*5.06e+05*	∞
Brown-headed Cowbird (*Molothrus ater*) *	80.4	173	*551*
Brown Thrasher (*Toxostoma rufum*)	19.3	*1.87e+03*	*53.5*
Carolina Chickadee (*Poecile carolinensis*) *	4.04	*30.8*	*16.5*
Carolina Wren (*Thryothorus ludovicianus*)	7.04	11.3	16.6
Cedar Waxwing (*Bombycilla cedrorum*)	*66.2*	*13.9*	*849*
Chimney Swift (*Chaetura pelagica*)	9.57	*26.6*	*28.6*
Chipping Sparrow (*Spizella passerina*)	13.5	18.5	*48*
Chuck-will's-widow (*Caprimulgus carolinensis*)	*2.65e+26*	∞	∞
Common Grackle (*Quiscalus quiscula*)	*34.5*	*34.8*	*56.5*
Common Nighthawk (*Chordeiles minor*)	14.8	*82*	*112*
Common Raven (*Corvus corax*)	6.97	*110*	*29.4*
Common Yellowthroat (*Geothlypis trichas*)	*38.6*	*25.3*	*76.1*
Dickcissel (*Spiza americana*)	17.7	*80.3*	*110*
Downy Woodpecker (*Picoides pubescens*)	25.5	∞	*1.35e+04*
Eastern Bluebird (*Sialia sialis*) *	3.23	*5.45*	5.48
Eastern Kingbird (*Tyrannus tyrannus*) *	12	*1.05e+04*	*35.8*
Eastern Meadowlark (*Sturnella magna*)	6.21	*7.97*	*9.75*
Eastern Phoebe (*Sayornis phoebe*)	*23.5*	*37.6*	*99.9*
Eastern Towhee (*Pipilo erythrophthalmus*)	11.8	21	38.7
Eastern Wood-Pewee (*Contopus virens*) *	5.61	*37.1*	*12.9*
European Starling (*Sturnus vulgaris*)	5.92	8.31	*15.6*
Field Sparrow (*Spizella pusilla*)	*7.94*	*30.1*	*21.3*
Grasshopper Sparrow (*Ammodramus savannarum*)	23.9	*2.29e+07*	*922*
Gray Catbird (*Dumetella carolinensis*)	*339*	*269*	*1.73e+03*
Great Blue Heron (*Ardea herodias*)	4.85	∞	*297*
Great Crested Flycatcher (*Myiarchus crinitus*) *	35.9	*2.7e+03*	*321*
Hermit Thrush (*Catharus guttatus*)	*7.44*	*22.5*	*23.8*
Horned Lark (*Eremophila alpestris*)	*7.71*	*35.3*	*35.2*
House Finch (*Carpodacus mexicanus*) *	34.4	*2.02e+06*	*647*
House Sparrow (*Passer domesticus*)	9.51	*19.6*	*30.6*

* Calibration hurt model performance.

continued on next page

Table 3.2: Data efficiency ratios for eBird compared to BBS *continued*.

SPECIES	AUC RATIO	ACC RATIO	RMS RATIO
House Wren (*Troglodytes aedon*) *	*33.5*	*592*	*352*
Indigo Bunting (*Passerina cyanea*)	9.84	8.66	18.1
Killdeer (*Charadrius vociferus*)	15	*94*	*70.8*
Lark Sparrow (*Chondestes grammacus*) *	*11.3*	*35.9*	*55.2*
Mourning Dove (*Zenaida macroura*)	22.4	*276*	*109*
Northern Bobwhite (*Colinus virginianus*)	11	16.8	31.9
Northern Cardinal (*Cardinalis cardinalis*)	14.9	12.7	26.2
Northern Mockingbird (*Mimus polyglottos*)	6.95	*11*	*13.4*
Orchard Oriole (*Icterus spurius*) *	14	∞	*439*
Ovenbird (*Seiurus aurocapilla*)	*13.1*	*25.9*	*54.9*
Pine Warbler (*Dendroica pinus*)	*4.66*	*41*	*15.8*
Purple Martin (*Progne subis*)	*133*	∞	*2.26e+04*
Red-bellied Woodpecker (*Melanerpes carolinus*) *	32.6	*148*	*153*
Red-eyed Vireo (*Vireo olivaceus*)	10.8	10.7	21.7
Red-tailed Hawk (*Buteo jamaicensis*)	*28.5*	*200*	*8.37e+03*
Red-winged Blackbird (*Agelaius phoeniceus*)	*28.5*	*55.9*	*102*
Ring-necked Pheasant (*Phasianus colchicus*)	12.9	*46*	*75.1*
Savannah Sparrow (*Passerculus sandwichensis*)	33.3	*3.45e+03*	*250*
Scarlet Tanager (*Piranga olivacea*) *	28.4	*132*	*134*
Song Sparrow (*Melospiza melodia*)	13.8	*26.9*	*45.6*
Spotted Towhee (*Pipilo maculatus*) *	26.6	*459*	*157*
Summer Tanager (*Piranga rubra*)	8.47	*279*	*74.8*
Tree Swallow (*Tachycineta bicolor*)	*70.5*	∞	*4.74e+03*
Tufted Titmouse (*Baeolophus bicolor*)	7.28	15.8	18.2
Veery (*Catharus fuscescens*)	10.1	*118*	*29.8*
Vesper Sparrow (*Pooecetes gramineus*)	10.1	*5.22e+03*	*169*
Warbling Vireo (*Vireo gilvus*) *	*22.1*	*1.34e+11*	*300*
Western Kingbird (*Tyrannus verticalis*)	*6.66*	*11.1*	*12.1*
Western Meadowlark (*Sturnella neglecta*)	6.53	*9.23*	*14.5*
White-breasted Nuthatch (*Sitta carolinensis*)	17.7	∞	*5.31e+03*
White-eyed Vireo (*Vireo griseus*)	5.08	*25.3*	*17.8*
Wood Thrush (*Hylocichla mustelina*)	12.4	*68.5*	*56.2*
Yellow-billed Cuckoo (*Coccyzus americanus*)	18	*182*	*92.8*
Yellow-breasted Chat (*Icteria virens*)	15.4	*858*	*346*
Yellow Warbler (*Dendroica petechia*)	20.9	*1.58e+03*	*178*

* Calibration hurt model performance.

Both the eBird and BBS maps differed from published range maps in field guides along the edges of ranges. We attribute this to the uncertainty of how to cartographically represent the lower limit of occurrence. For example, the models correctly predict the near zero-occurrence of western meadowlarks in the upper midwest of Indiana, Ohio, and Michigan (Fig. 3.5); some field guides include this area of low and patchy occurrence in the breeding range.

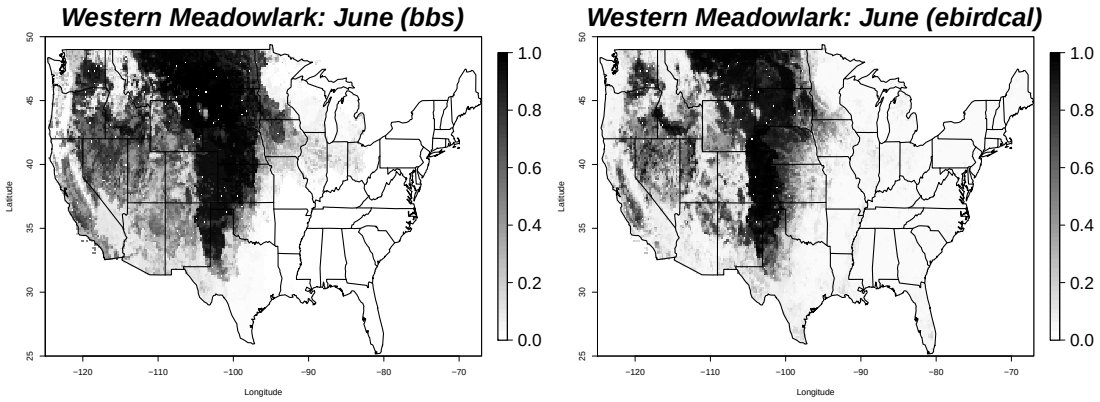

Figure 3.5: Western meadowlark (*Sturnella neglecta*) range boundaries and areas of concentration in the BBS and eBird maps are extremely similar and accurate. Western meadowlarks are widespread but declining grassland birds. Both maps correctly show lower frequency in high mountains (e.g., Rockies, Sierra Nevadas), but the eBird map reflects finer scale habitat distinctions in California, southeast Arizona, and western Colorado where meadowlarks occur only in the few patches of grassland and appropriate agriculture. eBird has better sampling coverage than BBS in California and southeast Arizona (Fig. 3.1), providing finer-scaled occurrence data. In western Colorado the BBS appears to have better coverage, but most routes follow open country roads in valleys and not windy mountain roads. Thus, mountainous habitat is under-sampled, biasing the data in the region.

3.4 Discussion

In this chapter we illustrate how the relative information content of data sets can be quantified for two different comparison goals. First, we show how the content of two data sets can be quantified for existing data. Second, we develop a more prospective comparison that asks how many data from one source would need to be collected to equal the information content of the reference source. Both methods are based on the assumption that a model linking predictor values to the response (e.g., species presence or abundance) is a good summary of the information content of the data used to create the model. We feel that this is intuitively appropriate, and our results—that data from the more-structured BBS protocol have a higher per-datum information content—is consistent with

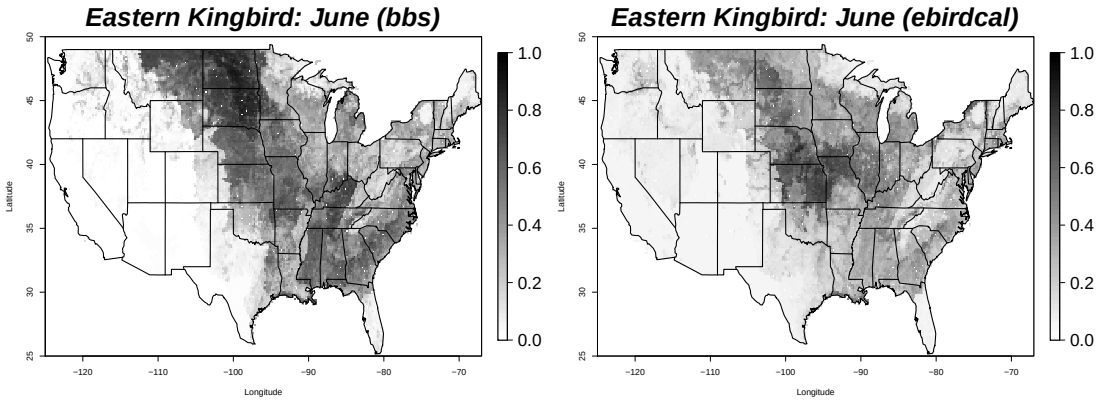

Figure 3.6: BBS and eBird maps of eastern kingbird (*Tyrannus tyrannus*) distribution differ mainly in their predicted occurrence rates, although the large-scale patterns are quite accurate for both. Eastern kingbird is a common bird of open country, roadsides, and pond edges with stable populations. It is not surprising that the BBS model predicts higher occurrence since it is exclusively a roadside survey. The maps agree on areas of concentration (central Great Plains and coastal Southeast), but the BBS map is more accurate for the Dakotas where the BBS coverage is more complete and uniform than for eBird (Fig. 3.1). Both maps overstate the lack of kingbirds in the Adirondacks of New York and northeastern Minnesota. While kingbirds do avoid heavily forested regions, kingbirds breed in these areas along lake edges and beaver ponds and in open area patches. The covariates used for modeling were likely measured at too coarse a resolution to detect these habitat distinctions.

our intuition. While we have used species distribution models in our comparisons, other forms of models (e.g., abundance distributions, population trends) could be substituted as appropriate. The most novel aspect of our work is the use of a cross-validation that validates a model using data from a different source than the training data (vs. traditional validation using independent data from the same source). Our approach allows us to isolate the information content of the data from possible overfitting of the collection protocol.

Unresolved issues do exist with this approach because each collection protocol has its own biases, and these biases are also reflected in the model built from the data. We made basic attempts to account for the differing biases of the two protocols we considered by using a simple calibration. As our results

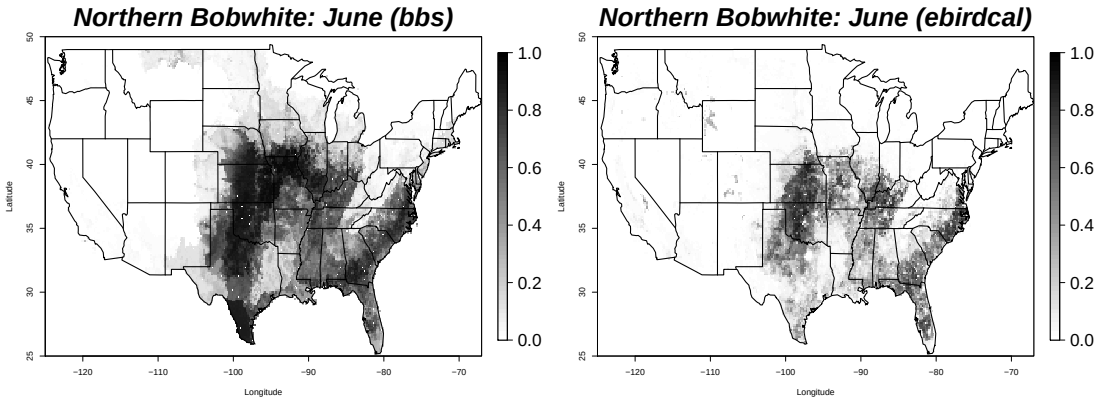

Figure 3.7: BBS and eBird maps of northern bobwhite (*Colinus virginianus*) correctly show the areas of high occurrence (Kansas, Oklahoma, and southeastern coastal plain) but both maps contain major mistakes elsewhere. Northern bobwhite is an eastern quail that has disappeared from the northern portions of its former range and whose populations continue to drastically decline range-wide. Bobwhites require sparsely populated farms and grasslands, and do not fare well in areas with people, dogs, and suburban development. eBird and BBS respectively under-sample and oversample bobwhite habitat, causing biases in both maps. eBird data are concentrated around cities and towns (Fig. 3.1) where bobwhites are absent, and the eBird model over-generalizes this pattern to large regions of near-zero occurrence. BBS routes favor rural countryside, and the BBS model overstates bobwhite occurrence near populated areas. The true distribution of bobwhites lies somewhere between these maps, with high occurrence in wild and agriculture areas (as in BBS map) and wide buffers of low occurrence around cities and suburban areas (as in eBird map).

show, more sophisticated approaches to addressing this issue are needed, as the calibration was not always effective. The failure patterns of simple calibration suggest that biases for a species vary by region as a function of the probability of occurrence; we suspect that the same issue would be encountered if modeling relative abundance instead of occurrence probabilities.

Another message from our results is that the modeling task (e.g., the targeted species and performance metric in our study) will alter the quantitative differences between data from different protocols. In our comparison, we see three possible explanations for why eBird compares most favorably to BBS for AUC, second most-favorably for ACC, and worst for RMS. First, ranking surveys from

most likely to see a particular species X to least likely is easier than predicting whether each survey actually did record X (AUC vs. ACC). Similarly, predicting if a survey recorded X is easier than guessing the exact probability of recording X (ACC vs. RMS). Second, AUC is unaffected by shifts in or (strictly monotone) scaling of detection probabilities. Third, tuning the threshold for ACC is easier than calibrating RMS, so there is less chance to introduce errors. Regardless of the actual reason(s), analysts will need to decide whether it is sufficient that data contain accurate information on ranking of occurrence rates (i.e. measure performance with AUCs), or whether absolute errors (measured by ACC or RMS) are important.

It is important to remember that the benchmark model represents both the biological signal and biases of the benchmark data. Differences between the biases of the data sources can easily prevent the candidate model from equaling the benchmark model's accuracy on independent benchmark data, regardless of how many data are available. We expect candidate model performance to often asymptote to a slightly worse level than benchmark performance, even when the performance trends are converging. Because the test data are biased, small discrepancies between candidate and benchmark models do not automatically imply that the candidate data contain less information; rather, the data sources are simply different. Either source could be slightly better, or they could be complementary. Determining which scenario is true is beyond the abilities of cross-data validation and remains a task for experts.

Regarding our specific comparison, we found that eBird-based and BBS-based models had similar predictive power, with eBird models being slightly less accurate than BBS models. The converging performance trends for 2/3 of the species for AUC suggest that the discrepancies between eBird and BBS—

at least for ranking sites by species suitability—will shrink as the volume of eBird data outstrips the volume of BBS data. By combining data efficiency ratios (Table 3.2) and the trends for eBird growth (Fig. 3.2), one can infer when enough eBird data will be collected to rival the information in BBS data for describing distributions. For example, the ACC data efficiency ratio for northern bobwhite is 16.8 with a converging trend. To collect as much information about bobwhite presence/absence as the BBS (annually), eBird needs to collect about 36,000 transect surveys. eBird's data volume increased 52% annually since 2003; at this pace, eBird will collect 41,000 breeding season transect surveys in 2010—enough to equal the information in the 2010 BBS surveys. In some cases, eBird data already describe species' distributions more accurately—as we found for western meadowlark, based on expert opinion.

Conversely, there are species for which eBird and BBS contain drastically different biases and sources of variance as evidenced by a few infinite data efficiency ratios in Table 3.2. For example, nocturnal birds like chuck-will's-widow (*Caprimulgus carolinensis*) are rarely counted in eBird (because most surveys start after dawn), yet are counted in BBS (because all BBS surveys start a half-hour before dawn). eBird data will never be comparable to BBS data in these cases (barring the development of methods to account for protocol biases), and experts should decide which data source is most appropriate to the goals of an analysis.

Although our analyses considered breeding season distributions, there is no reason to think eBird data collected during other seasons differs significantly in quality.

3.5 Conclusion

In conclusion, we believe that the methods described here can provide the basis for making decisions on appropriate choice of data to use, if a single source of data needs to be chosen for analysis. Alternatively, demonstration of reasonable information content in multiple data sources could open the door for learning from multiple data sets. For example, one could use one data set to construct a prior for Bayesian distribution models (Thogmartin & Knutson, 2007) that are then fit using the second data set (e.g., using eBird data to create informative priors for analysis of BBS data). Given the apparent consistency in informativeness of the data from the low-structure eBird protocol, the uses of such birder checklist data needs further exploration.

Cross-data validation implicitly requires that a learning algorithm can learn a suitable model for the task. In particular, the learned model should perform well according to performance measures that are relevant for the task. In this chapter we used ACC, RMS, and AUC to measure model performance. Many standard machine learning algorithms (and their common software implementations) optimize for ACC or RMS, but directly optimizing for AUC requires specialized algorithms. Chapter 4 studies an ensemble learning approach to optimizing for an *arbitrary* performance metric and evaluates its success in the domain of natural language processing.

CHAPTER 4

OPTIMIZING TO ARBITRARY NLP METRICS USING ENSEMBLE SELECTION *

There is measure in all things.

— Horace, *Satires* (Bartlett & Kaplan, 1992)

Most machine learning algorithms optimize for accuracy, squared error, or the likelihood of the training data (i.e., goodness of fit). In many applications of machine learning to real problems, however, success is not measured by the model's accuracy or squared error or goodness of fit. Instead, success is measured in terms of task specific performance metrics. The usefulness of machine learning in these domains depends critically on whether learning algorithms can be tuned to optimize for these alternative metrics.

This chapter evaluates an ensemble selection framework designed to optimize arbitrary metrics and automate the process of algorithm selection and parameter tuning. We study the method applied to the domain of natural language processing (NLP), a domain that heavily uses alternative performance measures. We report the results of experiments that instantiate the framework for three NLP tasks, using six learning algorithms, a wide variety of parameterizations, and 15 performance metrics. Based on our results, we make recommendations for subsequent machine-learning-based research for natural language learning.

*Large portions of this chapter were first published as:
Munson, A., Cardie, C., & Caruana, R. (2005). Optimizing to arbitrary NLP metrics using ensemble selection. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (eds. C. Brew, L.-F. Chien, & K. Kirchhoff), pp. 539–546. Association for Computational Linguistics, East Stroudsburg, PA, USA.

4.1 Introduction

Among the most successful natural language learning techniques for a wide variety of linguistic phenomena are supervised inductive learning algorithms for classification. Because of their capabilities for accurate, robust, and efficient linguistic knowledge acquisition, they have been employed in many NLP tasks.

Unfortunately, supervised classification algorithms are typically designed to optimize accuracy (e.g., decision trees) or mean squared error (e.g., neural networks). For many NLP tasks, however, these standard performance measures are inappropriate. For example, NLP data can be highly skewed in its distribution of positive and negative examples. In these situations, another metric (perhaps F-measure or a task-specific measure) that focuses on the performance of the minority cases is more appropriate. Indeed, as the NLP field matures more consideration will be given to evaluating the performance of NLP components in context (e.g., Is the system easy to use by end users? Does the component respect user preferences? How well does the entire system solve the specific problem?), leading to new and complicated metrics. Optimizing machine learning algorithms to arbitrary performance metrics, however, is not easily done.

To exacerbate matters, the metric of interest might change depending on how the natural language learning (NLL) component is employed. Some applications might need components with high recall, for example; others, high precision or high F-measure or low root mean squared error. To obtain good results with respect to the new metric, however, a different parameterization or different algorithm altogether might be called for, requiring re-training the classifier(s) from scratch.

Caruana et al. (2004) have recently proposed **ensemble selection** as a technique for building an ensemble of classifiers that is optimized to an arbitrary

performance metric. The approach trains a large number of classifiers using multiple algorithms and parameter settings, with the idea that at least some of the classifiers will perform well on any given performance measure. The best set of classifiers, with respect to the target metric, is then greedily selected. (Selecting a set of size 1 is equivalent to parameter and algorithm tuning.) Like other ensemble learning methods (e.g., bagging (Breiman, 1996) and boosting (Freund & Schapire, 1996)), ensemble selection has been shown to exhibit reliably better performance than any of the contributing classifiers for a number of learning tasks.

In addition, ensemble selection provides an ancillary benefit: no human expertise is required in selecting an appropriate machine learning algorithm or in choosing suitable parameter settings to get good performance. This is particularly attractive for the NLP community where researchers often rely on the same one or two algorithms and use default parameter settings for simplicity. Ensemble selection is a tool usable by non-experts to find good classifiers optimized to task-specific metrics.

This chapter evaluates the utility of the ensemble selection framework for NLL. We use three NLP tasks for the empirical evaluation: noun phrase coreference resolution and two problems from sentiment analysis — identifying private state frames and the hierarchy among them. The evaluation employs six learning algorithms, a wide selection of parameterizations, 8 standard metrics, and 7 task-specific metrics. Because ensemble selection subsumes parameter and algorithm selection, we also measure the impact of parameter and algorithm tuning.

Perhaps not surprisingly, we find first that no one algorithm or parameter configuration performs the best across all tasks or across all metrics. In ad-

dition, an algorithm's "tuned" performance (i.e., the performance after tuning parameter settings) almost universally matches or exceeds the algorithm's default performance (i.e., when using default parameter settings). Out of 154 total cases, the tuned classifier outperforms the default classifier 114 times, matches performance 28 times, and underperforms 12 times. Together, these results indicate the importance of algorithm and parameter selection for comparative empirical NLL studies. In particular, our results show the danger of relying on the same one or two algorithms for all tasks. These results cast doubt on conclusions regarding differences in algorithm performance for NLL experiments that give inadequate attention to parameter selection.

The results of our experiments that use ensemble selection to optimize the ensemble to arbitrary metrics show that ensemble selection usually improves performance but sometimes overfits. We see reliable improvements in performance across almost all of the metrics for two of the three tasks; for the other data set, ensemble selection tends to hurt performance (although losses are very small). Perhaps more importantly for our purposes, we find that ensemble selection provides small, but consistent gains in performance when considering only the more complex, task-specific metrics — metrics that learning algorithms would find difficult to optimize.

The rest of the chapter is organized as follows. Section 4.2 summarizes related work. Next, we describe the general learning framework and provide an overview of ensemble selection (§ 4.3). We then present the particular instantiation of the framework employed in our experiments (§ 4.4), describe the three NLP tasks (§ 4.5), and present the experimental results (§ 4.6). Conclusions follow in Section 4.7.

4.2 Related Work

4.2.1 Importance of Tuning Parameters

Hoste et al. (2002) and Hoste (2005) study the impact of tuning parameters for k-NN and a rule-learning algorithm on word sense disambiguation and coreference resolution, respectively, and find that parameter settings greatly change results. Similar work by Daelemans and Hoste (2002) shows the fallacy of comparing algorithm performance without first tuning parameters. They find that the best algorithm for a task frequently changes after optimizing parameters. In contrast to our work, these earlier experiments investigate at most two algorithms and only measure performance with one metric per task.

4.2.2 Optimizing Alternative Loss Functions

There are three approaches to learning models optimized for alternative loss functions. In the first approach learning algorithms are modified to accommodate each new performance metric. For example, a lot of research has studied how to make learning algorithms cost-sensitive. A **cost-sensitive** learner seeks to minimize the cost of the model's (incorrect) predictions instead of minimizing the number of incorrect predictions. Many domains associate different costs with different classification errors (e.g., medical diagnoses, fraud detection). Twenty-five years of cost-sensitive research have produced cost-sensitive decision trees (Breiman *et al.*, 1984; Turney, 1995), decision lists (Pazzani *et al.*, 1994), neural networks (Kukar & Kononenko, 1998), support vector machines (Lin *et al.*, 2002), and boosting (Ting & Zheng, 1998; Fan *et al.*, 1999), among others. Similarly, researchers have developed variant decision tree (Ferri *et al.*, 2002), neural network (Caruana *et al.*, 1996; Yan *et al.*, 2003; Herschtal & Raskutti,

2004), support vector machine (Herbrich *et al.*, 2000; Rakotomamonjy, 2004), and boosting (Freund *et al.*, 2003) algorithms that optimize for the area under the ROC curve (AUC). In the domain of information retrieval (i.e., finding documents relevant to a search query), neural networks and support vector machines have been extended to optimize for ranking performance measures like normalized discounted cumulative gain, mean average precision, and mean reciprocal rank (Burges *et al.*, 2007; Donmez *et al.*, 2009; Yue *et al.*, 2007). Directly optimizing for the desired loss function usually produces very good models, but modifying a learning algorithm requires significant expertise and the work must be repeated for each combination of learning algorithm and alternative loss function.[1]

The second approach relies on being able to learn a model that predicts good **class conditional probabilities** — that is, the probabilities that an example belongs to each of the possible classes — and then uses the probabilities to make a prediction that optimizes the desired performance metric. For example, a binary classification model that reliably predicts the probability of an example being positive or negative will also have a high AUC score. Many researchers have pursued this strategy for building cost-sensitive learning algorithms (e.g., Pazzani *et al.*, 1994; Bradford *et al.*, 1998; Zadrozny & Elkan, 2001). Unfortunately, predicting good probabilities is harder than optimizing most performance measures, and we do not currently have algorithms for learning well-calibrated models from high-dimensional data.

The final approach to optimizing for alternative loss functions is to reuse existing learning algorithms within a meta-algorithm that changes the inputs

[1]Occasionally a learning algorithm can be sufficiently generalized to allow optimizing for large classes of new alternative loss functions. A recent example is the multivariate support vector machine that can optimize for F measure, precision-recall break even point, precision at 10, area under the ROC curve, unbalanced classification costs, and any other loss function that can be computed from the confusion matrix — in addition to accuracy (Joachims, 2005).

to the existing algorithms and/or the outputs from the models. This strategy is similar to the probability approach above except that it avoids strong assumptions about the abilities of the existing learning algorithms. For example, Yang (2001) explored different threshold algorithms for converting model outputs to class predictions that optimize for F measure and the precision-recall break-even-point. His experiments focused on k nearest neighbor models, but the method could be applied to any learning algorithm. A popular strategy for cost-sensitive learning is to reweight or resample the training set to implicitly encourage the learning algorithm to focus on the most important examples (Ting, 1998; Elkan, 2001; Zadrozny *et al.*, 2003; Abe *et al.*, 2004). Lewis (2001) used the same reweighting trick to optimize for information retrieval performance measures. As a final example, the MetaCost algorithm (Domingos, 1999) uses bagging to generate better class conditional probabilities from base models. Based on these probabilities, the algorithm relabels training examples so that each example has the minimum cost label. A single cost-sensitive model is learned from the corrected training set.

The ensemble selection algorithm used here falls into this third category but is more general than the above approaches. Applying ensemble selection to a new performance measure does not require figuring out how to reweight training examples to optimize that measure. Similarly, the user does not need to decide how to modify or combine outputs from learned models; ensemble selection searches for a good combination method without any user input. As a result, ensemble selection is easier for practitioners to use.

4.3 Ensemble Selection Framework

4.3.1 Terminology

In the rest of this chapter we use the term model **model** to refer specifically to a classifier produced by some learning algorithm using some particular set of parameters. A model's **configuration** is simply the algorithm and parameter settings used to create the classifier. A **model family** is the set of models made by varying the parameters for one machine learning algorithm. Finally, a **model library** is a collection of models trained for a given task.

4.3.2 Framework

Abstractly, the framework is the following:

1. Select a variety of learning algorithms.
2. For each algorithm, choose a wide range of settings for the algorithm's parameters.
3. Divide data into training, tuning, and test sets.
4. Build model library.
5. Select target metrics appropriate to problem.
6. Tune parameter settings and/or run ensemble selection algorithm for target metrics.

Building the model library consists of (a) using the training data to train models for all the learning algorithms under all desired combinations of parameter settings, and (b) applying each model to the tuning and test set instances and storing the predictions for use in step (6). Note that models are placed in

the library regardless of performance, even though some models have very bad performance. Intuitively, this is because there is no way to know *a priori* which models will perform well on a given metric. Note that producing the base models is fully automatic and requires no expert tuning.

Parameter Tuning The goal of parameter tuning is to identify the best model for the task according to each target metric. Parameter tuning is handled in the standard way: for each metric, we select the model from the model library with the highest performance on the tuning data and report its performance on the test data.

Ensemble Selection Algorithm The ensemble selection algorithm (Caruana *et al.*, 2004) ignores model-specific details by *only using the predictions made by the models:* the ensemble makes predictions by averaging the predictions of its constituents. Advantageously, this only requires that predictions made by different models fall in the same range, and that they can be averaged in a meaningful way. Otherwise, models can take any form, including other ensemble methods (e.g., bagging or boosting). Conceptually, ensemble selection builds on top of the models in the library and uses their performance as a starting point from which to improve.

The basic ensemble selection algorithm is:

- Start with an empty ensemble.
- Add the model that results in the best performance for the current ensemble with respect to the tuning data and the target metric.
- Repeat (b) for a large, fixed number of iterations.
- The final ensemble is the ensemble from the best performing iteration on the tuning data for the target metric.

To prevent the algorithm from overfitting the tuning data we use two enhancements given by Caruana et al. (2004). First, in step (b) the same model can be selected multiple times (i.e., selection with replacement). Second, the ensemble is initialized with the top N models (again, with respect to the target metric on the tuning data). N is picked automatically such that removing or adding a model decreases performance on the tuning data.[2]

The main advantage to this framework is its reusability. After an instantiation of the framework exists, it is straightforward to apply it to multiple NLL tasks and to add additional metrics. Steps (1) and (2) only need to be done once, regardless of the number of tasks and metrics explored. Steps (3)-(5) need only be done once per NLL task. Importantly, the model library is created once for each task (i.e., each model configuration is only trained once) regardless of the number (or addition) of performance metrics. Finally, finding a classifier or ensemble optimized to a new metric (step (6)) does not require re-building the model library and is very fast compared to training the classifiers—it only requires averaging the stored predictions. For example, training the model library for our smallest data set took multiple days; ensemble selection built optimized ensembles for each metric in a few minutes.

4.4 Framework Instantiation

This section describes our instantiation of the ensemble selection framework.

Algorithms We use bagged decision trees (Breiman, 1996), boosted decision stumps (Freund & Schapire, 1996), k-nearest neighbor, a rule learner, and support vector machines (SVM). We use the following implementations of these

[2]We also experimented with the bagging improvement described by Caruana et al. (2004). In our experiments using bagging hurt the performance of ensemble selection.

algorithms, respectively: IND decision tree package (Buntine & Caruana, 1991); WEKA toolkit (Witten & Frank, 2000); TiMBL (Daelemans *et al.*, 2000); RIPPER (Cohen, 1995); and SVM^{light} (Joachims, 1999). Additionally, we use logistic regression (LR) for coreference resolution because an implementation using the MALLET (McCallum, 2002) toolkit was readily available for the task. The predictions from all algorithms are scaled to the range $[0, 1]$ with values close to 1 indicating positive predictions and values close to 0 indicating negative predictions.[3]

Parameter Settings Table 4.1 lists the parameter settings we vary for each algorithm. Additional domain-specific parameters are also varied for coreference resolution models (see Section 4.5.1). The model libraries contain models corresponding to the cross product of the various parameter settings for a given algorithm.

Standard Performance Metrics We evaluate the framework with 8 metrics: accuracy (ACC), average precision (APR), precision-recall break even point (BEP), F-measure (F1), mean cross entropy (MXE), root mean squared error (RMS), area under the ROC curve (AUC), and SAR. SAR (Caruana *et al.*, 2004) is defined as $SAR = \frac{ACC+AUC+(1-RMS)}{3}$. Precision and recall are commonly used metrics for natural language tasks and are used in computing F-measure, BEP, and APR.

[3]We follow Caruana et al. (2004) in using Platt scaling (Platt, 2000) to convert the SVM predictions from the range $(-\infty, \infty)$ to the required $[0, 1]$ by fitting them to a sigmoid. We also used Platt scaling to adjust the output from the boosted stumps since experiments by Niculescu-Mizil and Caruana (2005) indicate that Platt scaling calibrates the predictions made by boosting. We experimented with giving the same advantage to all of the models, i.e., using Platt scaling to adjust the output from all the algorithms. Indiscriminately scaling all the models hurt performance at least as often as it helped performance.

Table 4.1: Summary of model configurations used in experiments. The default settings for each algorithm are in bold.

ALGORITHM	PARAMETER	VALUES
Bagged Trees †	tree type	bayes, **c4**, cart, cart0, id3, mml, smml
	# bags	1, 5, 10, **25**
Boosted Stumps	# iterations	2, 4, 8, . . . , **256**, . . . 1024, 2048
LR ‡	gaussian gamma	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, **10**, 50
K-NN	k	**1**, 3, 5
	search algorithm	**ib1**, igtree
	similarity metric	**overlap**, modified value difference
	feature weighting	**gain ratio**, information gain, chi-squared, shared variance
Rule Learner	class learning order	unordered, **pos first**, neg first, heuristic determined order
	loss ratio	0.5, **1**, 1.5, 2, 3, 4
SVM	margin tradeoff*	10^{-7}, 10^{-6}, . . . , $\mathbf{10^{-2}}$, $\mathbf{10^{-1}}$, . . . , 10^{2}
	kernel	**linear**, rbf
	rbf gamma parm	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2

† Bagged trees are not used for identifying PSF's since the IND package does not support features with more than 255 values. Also, for coreference resolution the number of bags is not varied and is always 25.
‡ LR is only used for coreference resolution.
* SVM^{light} determines the default margin tradeoff based on data properties. We calculate this value for each data set and use the closest setting among our configurations.

They are defined as:

$$PRE = \frac{\text{\# correct pos predictions}}{\text{\# pos predictions}}$$
$$REC = \frac{\text{\# correct pos predictions}}{\text{\# pos in data}}$$

Our F measure places equal emphasis on precision and recall, *i.e.* $\beta = 1$ in

$$F = \frac{(\beta^2 + 1) \times PRE \times REC}{\beta^2 \times PRE + REC}$$

Note that precision and recall are calculated *with respect to the positive class*. We also evaluate the effects of model selection with task-specific metrics. These are described in Section 4.5.

Ensemble Selection For the sentiment analysis tasks, ensemble selection iterates 150 times; for the coreference task, the algorithm iterates 200 times. This

should be enough iterations, given that the model libraries contain 202, 173, and 338 models. Because computing the MUC-F1 and BCUBED metrics (see Section 4.5.1) is expensive, ensemble selection only iterates 50 times for these metrics.

4.5 Tasks

This section briefly summarizes each NLL task. Readers are referred to the cited papers to obtain detailed descriptions.

4.5.1 Noun Phrase Coreference Resolution

The goal for a standard noun phrase coreference resolution system is to identify the noun phrases in a document and determine which of them refer to the same entity. Entities can be people, places, things, *etc*. The resulting partitioning of noun phrases creates reference chains with one chain per entity.

We use the same problem formulation as Soon et al. (2001) and Ng and Cardie (2002) — a combination of classification and clustering. Briefly, every noun phrase is paired with all preceding noun phrases, creating multiple pairs. For the training data, the pairs are labeled as coreferent or not. A binary classifier is trained to predict the pair labels. During classification, the predicted labels are used to form clusters. Two noun phrases A and B share a cluster if they are either predicted as coreferent by the classifier or if they are transitively predicted as coreferent through one or more other noun phrases. Instance selection (Soon *et al.*, 2001; Ng, 2004) is used to increase the percentage of positive instances in the training set.[4]

[4] `Soon-1` instance selection is used for all algorithms; we also use `soon-2` (Ng, 2004) instance selection for the rule learner.

We use the learning features described by Ng and Cardie (2002). All learning algorithms are trained with the full set of features. Additionally, the rule learner, SVM, and LR are also trained with a hand-selected subset of the features that Ng and Cardie (2002) find to outperform the full feature set. Essentially this is an additional parameter to set for the learning task (i.e., train with all features or with the hand-selected subset).

Special Metrics Rather than focusing on performance at the pairwise coreference classification level, performance for this task is typically reported using either the MUC metric (Vilain *et al.*, 1995) or the BCUBED metric (Bagga & Baldwin, 1998). Both of these metrics measure the degree that predicted coreference chains agree with an answer key. In particular they measure the chain-level precision and recall (and the corresponding F-measure). We abbreviate these metrics MUC-F1, and B3F1.

Data Set For our experiments we use the MUC-6 corpus, which contains 60 documents annotated with coreference information. The training, tuning, and test sets consist of documents 1-20, 21-30, and 31-60, respectively.

4.5.2 Identifying Private State Frames

NLP research has recently started looking at how to detect and understand subjectivity in discourse. A key step in this direction is automatically identifying phrases that express subjectivity. In this setting, subjectivity is defined to include implicit and explicit opinions, internal thoughts and emotions, and bias introduced through second-hand reporting. Phrases expressing any of these are called **private state frames**, which we will abbreviate as PSF.

We build directly on experiments done by Wiebe et al. (2003). The task is to learn to identify explicit single-word PSF's in context. One learning instance is created for every word in the corpus. Classification labels each instance as a PSF or not. We use the same features as Wiebe et al.

Special Metrics Because the data is highly skewed (2% positive instances), performance measures that focus on how well the minority class is learned are of primary interest. The F-measure defined in Section 4.4 is a natural choice. We also evaluate performance using geometric accuracy, defined as

$$GACC = \sqrt{posacc \times negacc} \tag{4.1}$$

where $posacc$ and $negacc$ are the accuracy with respect to the positive and negative instances (Kubat & Matwin, 1997).

Conceivably, an automatic PSF extractor with high precision and mediocre recall could be used to automate the annotation process. For this reason we measure the performance with an unbalanced F-measure that emphasizes precision. Specifically, we try $\beta = 0.5$ (F0.5) and $\beta = 0.2$ (F0.2).

Data Set We use 400 documents from the MPQA corpus (MPQA Corpus, 2002), a collection of news stories manually annotated with PSF information. The 400 documents are randomly split to get 320 training, 40 tuning, and 40 testing documents.

4.5.3 Determining PSF Hierarchy

The third task is taken from Breck and Cardie (2004). Explicit PSFs each have a **source** that corresponds to the person or entity expressing the subjectivity. In the presence of second-hand reporting, sources are often nested. This has the effect

of filtering subjectivity through a chain of sources. Breck and Cardie (2004) give the following example sentence (PSFs are in bold, sources are in italics):

Alice's **claim** that *Bob* was **unhappy angered** *Charlie*.

There are three source-PSF pairs in the example: Alice's claim, Bob's unhappiness, and Charlie's anger. Bob's unhappiness is filtered through Alice's claim. Implicitly, both Alice's claim and Charlie's anger are filtered by the sentence's writer. Combining these relationships creates a hierarchical tree structure. The ability to determine hierarchical structure among subjective expressions is believed to be important for summarizing opinions and for answering multi-perspective questions.

Given sentences annotated with PSF information (i.e., which spans are PSFs), the task is to discover the hierarchy among the PSFs that corresponds to the nesting of their respective sources. From each sentence, multiple instances are created by pairing every PSF with every other PSF in the sentence.[5] Let $(PSF_{parent}, PSF_{target})$ denote one of these instances. The classification task is to decide if PSF_{parent} is the parent of PSF_{target} in the hierarchy and to associate a confidence with that prediction. The complete hierarchy can easily be constructed from the predictions by choosing for each PSF its most confidently predicted parent.

Special Metrics Breck and Cardie (2004) measure task performance with three metrics. The first is the accuracy of the predictions over the instances. The second is a derivative of a measure used to score dependency parses. Essentially, a sentence's score is the fraction of parent links correctly identified. The score for a set of sentences is the average of the individual sentence scores. We refer

[5]Sentences containing fewer than two PSF's are discarded and not used. Every PSF is also paired with the implicit writer source that is the root of every hierarchy.

to this measure as average sentence accuracy (SENTACC). The third measure is the percentage of sentences whose hierarchical structures are perfectly determined (PERFSENT).

Data Set We use the same data set and features as Breck and Cardie (2004). The annotated sentences from 469 documents in the MPQA Corpus (2002) are randomly split into training (80%), tuning (10%), and test (10%) sets.

4.6 Experiments and Results

We evaluate the potential benefits of the ensemble selection framework with two experiments. The first experiment measures the performance improvement yielded by parameter tuning and finds the best performing algorithm. The second experiment measures the performance improvement from ensemble selection.

Performance improvements are measured in terms of performance **gain**. Let a and b be the measured performances for two models A and B on some metric. A's gain over B is simply $a - b$. A performed worse than B if the gain is negative.[6]

4.6.1 Experiment 1: Parameter Tuning

Experiment 1 measures, for each of the 3 tasks, the performance of every model on both the tuning and test data for every metric of interest. *Based on tuning set performance,* the best default model, the best model overall, and the best model

[6]MXE and RMS have inverted scales where the best performance is 0. Gain for these metrics equals $(1 - a) - (1 - b)$ so that positive gains are always good. Similarly, where raw MXE and RMS scores are reported we show $1 - score$.

within each family are selected. The **best default model** is the highest-scoring model that emerges after comparing algorithms without doing any parameter tuning. The **best overall model** corresponds to "tuning" both the algorithm and parameters. The **best model in each family** corresponds to "tuning" the parameters for that algorithm. *Using the test set performances,* the best family models are compared to the corresponding default models to find the gains from parameter tuning.

Table 4.2 lists the gains achieved by parameter tuning. Each algorithm column compares the algorithm's best model to its default model. On the coreference task, for example, the best KNN model with respect to BEP shows a 20% improvement (or gain) over the default KNN model (for a final BEP score of .5300). The "Avg Δ" column shows the average gain from parameter tuning for each metric.

For each metric, the best default model is italicized while the best overall model is bold-faced. Referring again to the coreference BEP row, the best overall model is a SVM while the best default model is a bagged decision tree. Recall that these distinctions are based on *absolute performance* and not gain. Thus, the best tuned SVM outperforms all other models on this task and metric.[7]

Three conclusions can be drawn from Table 4.2. *First, no algorithm performs the best on all tasks or on all metrics.* For coreference, the best overall model is either a bagged tree, a rule learner, or a SVM, depending on the target metric. Similarly, for PSF identification the best model depends on the metric, ranging from a KNN to a SVM. Interestingly, bagged decision trees on the PSF hierarchy data outperform the other algorithms on all metrics and seem especially well-

[7] Another interesting example is the best overall model for BEP on the PSF hierarchy task. The baseline (a bagged tree) outperforms the "best" model (a different bagged tree) on the test set even though the best model performed better on the tuning set—otherwise it would not have been selected as the best.

Table 4.2: Performance gains from parameter tuning. The left column for each algorithm family is the algorithm's performance with default parameter settings. The adjacent 'ParmΔ' column gives the performance gain from tuning parameters. For each metric, the best default and tuned performance across all algorithms are *italicized* and **bold-faced**, respectively.

	Metric	BAG	ParmΔ	BST	ParmΔ	LR	ParmΔ	KNN	ParmΔ	RULE	ParmΔ	SVM	ParmΔ	Avg Δ
coreference	ACC	0.9861	-0.0000	*0.9861*	-0.0001	0.9849	0.0006	0.9724	0.0131	0.9840	**0.0023**	0.9859	-0.0001	0.0026
	APR	*0.5373*	**0.0000**	0.5475	0.0000	0.3195	-0.0004	0.1917	0.2843	0.2491	0.1127	0.5046	0.0323	0.0715
	AUC	0.9258	**0.0158**	*0.9466*	0.0000	0.4275	0.0022	0.7746	0.0954	0.6845	0.1990	0.8418	0.0551	0.0612
	BEP	*0.6010*	0.0000	0.5577	0.0193	0.3747	-0.0022	0.3243	0.2057	0.3771	0.1862	0.5965	**0.0045**	0.0689
	F1	*0.5231*	**0.0664**	0.3881	0.0000	0.4600	0.0087	0.4105	0.1383	0.4453	0.1407	0.3527	0.0571	0.0685
	MXE	0.9433	**0.0082**	*0.9373*	0.0000	0.5400	0.1828	0.4953	0.3734	0.9128	0.0222	0.9366	0.0077	0.0990
	RMS	0.8925	0.0041	*0.8882*	0.0000	0.6288	0.1278	0.8334	0.0559	0.8756	0.0097	0.8859	**0.0047**	0.0337
	SAR	0.9255	**0.0069**	*0.9309*	0.0000	0.6736	-0.0037	0.8515	0.0538	0.8396	0.0695	0.8955	0.0165	0.0238
	MUC-F1	*0.6691*	**0.0000**	0.6242	0.0046	0.6405	0.0344	0.5340	0.1185	0.6500	0.0291	0.5181	0.1216	0.0514
	B3F1	*0.4625*	0.0000	0.4512	0.0000	0.4423	0.0425	0.0965	0.3357	0.4249	**0.0675**	0.3323	0.1430	0.0981
PSF identification	ACC	—	—	0.9854	0.0007	—	—	0.9873	0.0011	0.9862	0.0003	*0.9886*	**0.0000**	0.0005
	APR	—	—	0.6430	0.0316	—	—	0.5588	0.1948	0.4335	0.0381	*0.7697*	**0.0372**	0.0754
	AUC	—	—	0.9576	0.0121	—	—	0.8566	0.1149	0.7181	0.1593	*0.9659*	**0.0188**	0.0763
	BEP	—	—	0.5954	0.0165	—	—	0.6727	0.0302	0.4436	0.0718	*0.6961*	**0.0385**	0.0393
	F1	—	—	0.5643	0.0276	—	—	0.6837	0.0019	0.5770	0.0367	*0.6741*	**0.0062**	0.0181
	MXE	—	—	0.9342	0.0029	—	—	0.8089	0.1425	0.9265	0.0062	*0.9572*	**0.0093**	0.0402
	RMS	—	—	0.8838	0.0028	—	—	0.8896	0.0118	0.8839	0.0020	*0.9000*	**0.0068**	0.0058
	SAR	—	—	0.9329	0.0052	—	—	0.9021	0.0407	0.8541	0.0532	*0.9420*	**0.0085**	0.0269
	GACC	—	—	0.6607	-0.0004	—	—	*0.7962*	**0.0223**	0.6610	0.0506	0.7401	0.0209	0.0233
	F0.5	—	—	0.6829	0.0221	—	—	0.7150	0.0503	0.7132	0.0000	*0.7811*	**-0.0054**	0.0167
	F0.2	—	—	0.7701	0.0157	—	—	0.7331	0.0875	*0.8171*	**0.0110**	0.8542	0.0045	0.0297
PSF hierarchy	ACC	*0.8133*	**0.0000**	0.7554	0.0009	—	—	0.7940	0.0000	0.7446	0.0428	0.7761	0.0381	0.0164
	APR	0.8166	**0.0296**	0.7455	0.0013	—	—	*0.8035*	0.0000	0.5957	0.1996	0.6363	0.1520	0.0765
	AUC	*0.8923*	**0.0096**	0.8510	0.0000	—	—	0.8519	0.0364	0.7514	0.1094	0.7968	0.0757	0.0462
	BEP	*0.7385*	**-0.0066**	0.6597	-0.0030	—	—	0.7096	0.0000	0.6317	0.0567	0.6940	0.0432	0.0181
	F1	*0.7286*	**0.0033**	0.6810	0.0226	—	—	0.7000	0.0000	0.6774	0.0525	0.6933	0.0400	0.0237
	MXE	*0.6091*	**0.0166**	0.4940	0.0076	—	—	0.0379	0.4715	0.4022	0.1197	0.4681	0.1012	0.1433
	RMS	*0.6475*	**0.0054**	0.5910	0.0033	—	—	0.6057	0.0000	0.5556	0.0514	0.5836	0.0423	0.0205
	SAR	*0.7765*	**0.0073**	0.7251	0.0009	—	—	0.7430	0.0000	0.6770	0.0672	0.7116	0.0482	0.0247
	SENTACC	*0.7571*	**0.0045**	0.7307	-0.0011	—	—	0.7399	-0.0007	0.6801	0.0141	0.6889	0.0726	0.0179
	PERFSENT	*0.4948*	**0.0069**	0.4880	0.0000	—	—	0.4880	-0.0034	0.4055	0.0206	0.4158	0.1031	0.0254

suited to the task.

Second, an algorithm's best-tuned model reliably yields non-trivial gains over the corresponding default model. This trend appears to hold regardless of algorithm, metric, and data set. In 114 of the 154 cases parameter tuning improves an algorithm's performance by more than 0.001 (0.1%). In the remaining 40 cases, parameter tuning only hurts 12 times, and never by more than 0.01.

Third, the best default algorithm is not necessarily the best algorithm after tuning parameters. The coreference task, in particular, illustrates the potential problem with using default parameter settings when judging which algorithm is most suited for a problem: 7 out of 10 times the best algorithm changes after parameter tuning.

These results corroborate those found elsewhere (Daelemans & Hoste, 2002; Hoste *et al.*, 2002; Hoste, 2005)—parameter settings greatly influence performance. Further, algorithmic performance differences can change when parameters are changed. Going beyond previous work, our results also underline the need to consider multiple algorithms for NLL. Ultimately, it is important for researchers to thoroughly explore options for **both** algorithm and parameter tuning and to report these in their results.

4.6.2 Experiment 2: Ensemble Selection

In experiment 2 an ensemble is built to optimize each target metric. The ensemble's performance is compared to that of the best overall model for the metric. Both the ensemble and the best model are selected according to tuning set performance.

Table 4.3 lists the gains from ensemble selection over the best parameter tuned model. For comparison, the best default and overall performances from

Table 4.3: Impact from tuning and ensemble selection. *Best default* shows the performance of the best classifier with no parameter tuning (*i.e.* algorithm tuning only). *Best tuned*Δ gives the performance gain from parameter and algorithm tuning. *Ens. Sel.*Δ is the performance gain from ensemble selection over the best tuned model. The best performance for each metric is marked in bold.

	Metric	**Best Default**	**Best TunedΔ**	**Ens. Sel.Δ**
coreference	ACC	0.9861	0.0002	**0.0001**
	APR	0.5373	0.0000	**0.0736**
	AUC	0.9466	-0.0051	**0.0120**
	BEP	0.6010	0.0000	**0.0124**
	F1	0.5231	0.0664	**0.0115**
	MXE	0.9373	0.0142	**0.0035**
	RMS	0.8882	0.0023	**0.0049**
	SAR	0.9309	0.0015	**0.0032**
	MUC-F1	0.6691	0.0000	**0.0073**
	B3F1	0.4625	0.0299	**0.0077**
PSF identification	ACC	0.9886	0.0000	**0.0003**
	APR	0.7697	0.0372	**0.0109**
	AUC	0.9659	0.0188	**0.0043**
	BEP	0.6961	0.0385	**0.0136**
	F1	0.6741	0.0062	**0.0222**
	MXE	0.9572	0.0093	**0.0029**
	RMS	0.9000	0.0068	**0.0025**
	SAR	0.9420	0.0085	**0.0021**
	GACC	0.7962	**0.0223**	-0.0012
	F0.5	0.7811	-0.0054	**0.0063**
	F0.2	0.8171	0.0110	**0.0803**
PSF hierarchy	ACC	**0.8133**	**0.0000**	-0.0028
	APR	0.8035	**0.0427**	-0.0064
	AUC	0.8923	**0.0096**	-0.0036
	BEP	**0.7385**	-0.0066	0.0056
	F1	0.7286	**0.0033**	-0.0016
	MXE	0.6091	**0.0166**	-0.0012
	RMS	0.6475	**0.0054**	-0.0019
	SAR	0.7765	**0.0073**	-0.0015
	SENTACC	0.7571	0.0045	**0.0024**
	PERFSENT	0.4948	0.0069	**0.0172**

Table 4.2 are reprinted. For example, the ensemble optimized for F1 on the coreference data outperforms the best bagged tree model by about 1% (and the best default model by almost 8%).

Disappointingly, ensemble selection does not consistently improve performance. Indeed, for the PSF hierarchy task ensemble selection reliably hurts performance a small amount. For the other two tasks ensemble selection reliably improves all metrics except GACC (a small loss). In other experiments, however, we optimized F-measure with $\beta = 1.5$ for the PSF identification task. Ensemble selection hurt F1.5 by almost 2%, leading us to question the technique's reliability for our second data set. Interestingly, the aggregate metrics—metrics that measure performance by combining multiple predictions—all benefit from ensemble selection, even for the hierarchy task, *albeit* for small amounts. For our tasks these comprise a subset of the task-specific performance measures: B3F1, MUC-F1, SENTACC, and PERFSENT.

Figure 4.1 depicts the *relative* improvements from parameter tuning and ensemble selection when compared to baseline performance (the best default model, i.e., algorithm tuning). Relative gains are computed as:

$$\text{rel. gain} = \left(1 - \frac{1 - \text{perf}}{1 - \text{baseline perf}}\right) \times 100 \tag{4.2}$$

For example, if the baseline has 80% accuracy and the best model (after parameter tuning) has 90% accuracy, the relative gain from parameter tuning is 50%. Relative gains illustrate the relative importance of the absolute differences in Tables 4.2 and 4.3.

These graphs show that improvements from ensemble selection, compared to parameter tuning, are usually relatively small (although the improvement for F0.2 on the PSF identification task is huge). Where ensemble selection performs worse than parameter tuning, the relative losses are small except for APR and

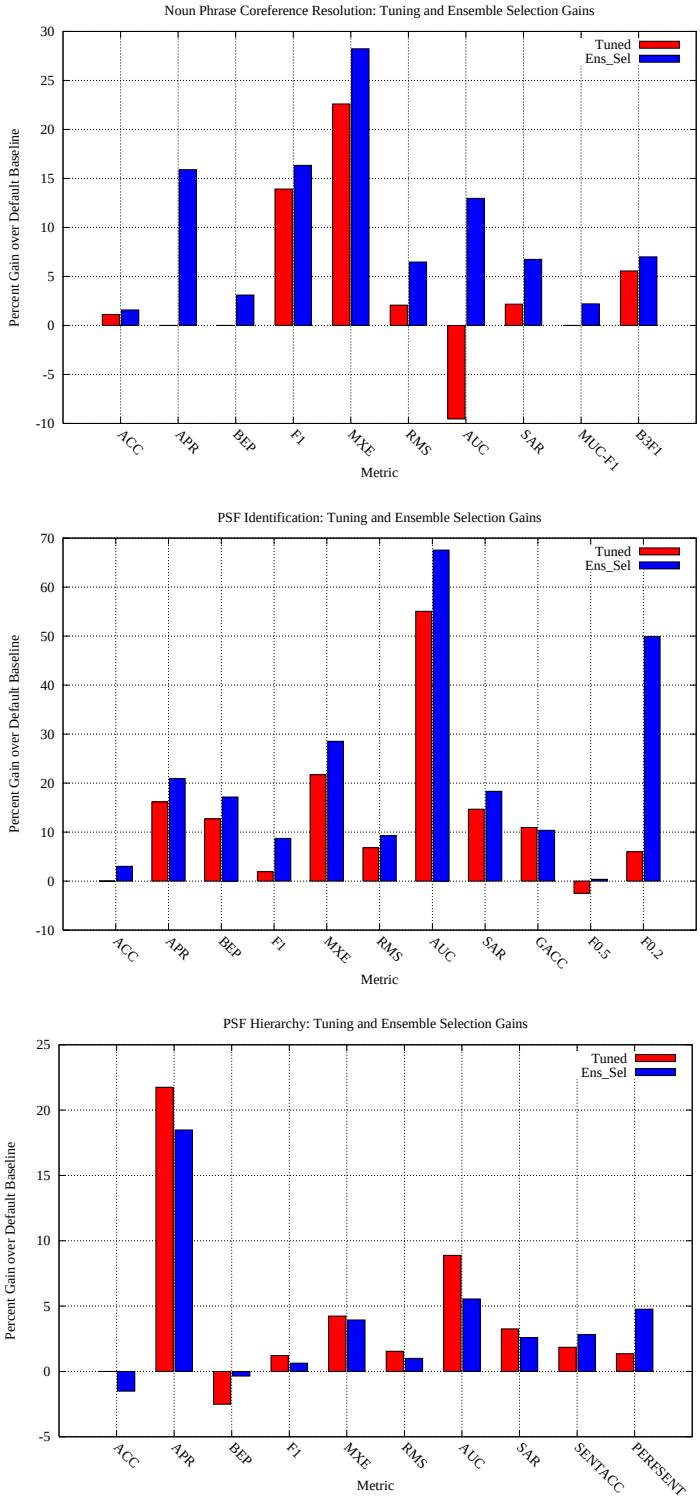

Figure 4.1: Relative gains from parameter tuning and ensemble selection. Zero represents the performance of the baseline (i.e., best single model using default parameter settings). Bars show relative gains (or losses) after normalizing by baseline performance.

AUC on the PSF hierarchy task.

While we are not surprised that the positive gains are small,[8] we are surprised at how often ensemble selection hurts performance. As a result, we investigated some of the metrics where ensemble selection hurts performance and found that ensemble selection overfits the tuning data. At this time we are not sure why this overfitting happens for these tasks and not for the ones used by Caruana et al. Preliminary investigations suggest that having a smaller model library is a contributing factor (Caruana et al. use libraries containing ~ 2000 models). This is consistent with the fact that the task with the largest model library, coreference, benefits the most from ensemble selection. Perhaps the reason that ensemble selection consistently improves performance for the aggregate metrics is that these metrics are harder to overfit.

Based on our results, ensemble selection seems too unreliable for general use in NLL—at least until the model library requirements are better understood. However, ensemble selection is perhaps trustworthy enough to optimize metrics that are difficult to overfit and could not be easily optimized otherwise — in our case, the task-specific aggregate performance measures.

4.7 Conclusion

We evaluated an ensemble selection framework that enables optimizing classifier performance to arbitrary performance metrics without re-training. An important side benefit of the framework is the fully automatic production of base-level models, removing the need for human expertise in choosing algorithms and parameter settings.

[8] Caruana et al. (2004) find the benefit from ensemble selection is only half as large as the benefit from carefully optimizing and selecting the best models in the first place.

Our experiments show that ensemble selection, compared to simple algorithm and parameter tuning, reliably improves performance for six of the seven task-specific metrics and all four "aggregate" metrics, but only benefits *all* of the metrics for one of our three data sets. We also find that exploring multiple algorithms with a variety of settings is important for getting the best performance. Tuning parameter settings results in 0.05% to 14% average improvements, with most improvements falling between 2% and 10%. To this end, the ensemble selection *framework* can be used as an environment for automatically choosing the best algorithm and parameter settings for a given NLP classification task. More work is needed to understand when ensemble selection can be safely used for NLL.

Looking back at Tables 4.2 and 4.3, one finds that the best model for any task or metric was almost always an ensemble of some kind (either a bagged decision tree model or an ensemble built using ensemble selection). This pattern agrees with findings in the machine learning community that ensemble learning is one of the best learning strategies we currently have (Caruana & Niculescu-Mizil, 2006; Caruana *et al.*, 2008). Unfortunately, ensembles are also difficult models to study and understand. Chapter 5 studies how to understand a particular type of ensemble: ensembles of decision trees.

CHAPTER 5

UNDERSTANDING DECISION TREE ENSEMBLES *

> *My biostatistician friends tell me, "Doctors can interpret logistic regression." There is no way they can interpret a black box containing fifty trees hooked together. In a choice between accuracy and interpretability, they'll go for interpretability.*
>
> *. . . The point of a model is to get useful information about the relation between the response and the predictor variables. Interpretability is a way of getting information. But a model does not have to be simple to provide reliable information about the relation between predictor and response variables.*
>
> — Leo Breiman (2001b, pp. 209–210)

Opaque models—that is, models that cannot be examined and understood—are unacceptable for many applications. While ensemble learning methods consistently learn high performing models from data, the resulting models are hard to understand because predictions are made by combining base models—often hundreds or even thousands of models. This chapter describes efficient statistics for identifying the most important predictors used by an ensemble of decision trees. The work grew out of a collaboration with the Cornell Lab of Ornithology. The biologists were happy with how accurately bagged decision tree models predicted the occurrence of bird species but really wanted to know how the models made predictions. In other words, what domain knowledge had the models extracted from the data to produce such good accuracy?

The first part of this chapter motivates the need for these fast statistics and describes the statistics. The second part of the chapter applies the statistics to

*This chapter is an expanded version of work published as:
Caruana, R., Elhawary, M., Fink, D., Hochachka, W.M., Kelling, S., Munson, A., Riedewald, M., Sorokina, D. (2006). Mining citizen science data to predict prevalence of wild bird species. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. T. Eliassi-Rad, L. Ungar, M. Craven, & D. Gunopulos), pp. 909–915. ACM, New York, NY, USA.

understanding models of wintertime avian populations and empirically shows that the statistics correlate well with sensitivity analysis (Breiman, 2001a), a black box method for measuring predictor importance that does not scale well enough to be generally applied to avian distribution models. The final part of the chapter applies partial dependence functions (Friedman, 2001) to visualize how the frequency of bird occurrences depends on the identified important predictors, and discusses the limits of identifying important predictors when predictors are correlated.

5.1 Motivation

Ecology is fundamentally the science of understanding the distribution and abundance of organisms. Ecologists interested in efficient environmental manipulation for conservation and management of wild birds have two general needs: (1) to be able to accurately predict where a species is and is not found; and (2) to understand the causes of presence and absence of a species. Within ecology, the conventional paradigm for analyzing data and gaining insights has been to start with the simplest model that might apply (based on domain knowledge) and then incrementally elaborate the model by adding new predictors (often one at a time) until the model is sufficiently accurate. Hypothesis testing is used to choose between competing models (if more than one model seems like a reasonable description of the biological processes at work, again based on domain knowledge). Models produced through this conventional process are only as complex as necessary because of how scientists build them and because of the human time needed to analyze and expand them. As a result, these models are

intrinsically understandable.[1]

This conventional paradigm is now becoming unworkable, overwhelmed by increasingly available large ornithology data sets with many potentially important predictors (e.g., geographic data sets based on satellite imagery). One example of this is the Avian Knowledge Network (AKN)[2], a group of university, governmental, and non-governmental ornithological organizations that are combining their existing databases of bird distribution information. Currently, almost 74 million bird observation records exist in the AKN's data warehouse, each record associated with more than 200 environmental predictors. This volume of data requires new scalable analytical tools that provide ecologists with initial insights (hypotheses) to be subsequently examined in greater detail.

Working with data from Project FeederWatch (PFW; see Section 5.4.1 for details of the data), we tried many different machine learning algorithms to learn models from large-scale, noisy ecological data. The best performing models were bagged decision trees. In addition to yielding good performance, the tree ensembles were able to gracefully handle the missing values in the PFW data. Unfortunately, although each individual tree could be examined and understood, the ensemble models—which contained 100 trees—were opaque black boxes. How can one understand the relationship between inputs and the model output when the output is derived from 100 subcomponents (base models) that all use the inputs in different ways?

One question worth asking is whether ensembles are actually necessary for this application; maybe a single decision tree could be used instead without sacrificing too much predictive accuracy. After all, a single tree is readily examinable. Unfortunately the performance of a single decision tree is substantially

[1] Indeed, it is hard to imagine an expert hand building a model that he or she cannot understand.

[2] `http://avianknowledge.net`

worse than that of a bagged tree ensemble for this domain (Table 5.1).

Table 5.1: Single decision trees perform substantially worse than bagged decision trees at predicting the winter time occurrence of house finches. Even more sophisticated tree types (e.g., BAYES, CART, C4.5, MML, SMML) are significantly worse. (See section 2.5 for an explanation of the various tree types.) Details about the prediction task are in section 5.4.1.

MODEL	ACC	RMS	AUC
bagged trees (100 ID3 trees)	0.826	0.354	0.894
single ID3 tree	0.786	0.425	0.818
single BAYES tree	0.793	0.389	0.858
single CART tree	0.797	0.391	0.844
single C4.5 tree	0.765	0.407	0.842
single MML tree	0.804	0.375	0.867
single SMML tree	0.736	0.422	0.792

Since ensembles are necessary to get the best performance, our goal is to find the predictors that are most important to the model's success and then visualize the relationships between them and species occurrence (i.e., the model outputs).

Sensitivity analysis (Breiman, 2001a, pp. 23–25) is a technique for measuring the importance of predictors to a model's predictions. The idea is to compare the performance of the model on a test set before and after noise is added to the target predictor. To measure the importance of predictor A, all A-values are shuffled, essentially permuting the original vector of A-values (when viewing the data set as a matrix whose rows are the different observation records and columns correspond to the different predictors). If the predictor is important, performance should drop on the perturbed test data set compared to the real one, because the model relies on the spoiled values when making predictions. Sensitivity analysis is a **black box method**: it does not rely on the model's structure and consequently can be applied to any model.

Sensitivity analysis is a relatively fast method for estimating variable impor-

tance. Once the model is trained, we only need to evaluate its performance for different perturbed test data sets, one for each predictor. This is much faster than the costly approach of re-training models for different sets of predictor variables, as required for variable selection methods (Kira & Rendell, 1992; Kohavi & John, 1997; Guyon & Elisseeff, 2003a). Nevertheless, for large high-dimensional data sets like PFW, even sensitivity analysis requires considerable resources: evaluating the sensitivity of a *single* predictor using 32K test cases takes about 4-5 minutes.[3] Using this approach for all 197 predictors of interest (or even pairs or larger sets of predictors) and for all 600+ region-species combinations requires access to expensive high-performance computing resources. Asymptotically, sensitivity analysis is $\Theta(npt)$, where n is the number of test cases, p is the number of predictors studied, and t is the time for the model to make a single prediction. Prediction time for an ensemble model increases with the number of base models in the ensemble and can be significant for large ensembles; i.e.,

$$t = \sum_{i=1}^{b} t_i$$

where b is the number of base models and t_i is the time for the ith model to make a prediction.

Our work proposes new statistics for predictor importance that exploit the structure of decision trees. As a result, they are much faster than sensitivity analysis but are only applicable to tree-based models.

[3]Time estimates based on single processor PC (3.6GHz, 1GB RAM).

5.2 Prior Work on Understanding Models

Most previous work on understanding complex models can be categorized into four strategies: understandable by design, complexity penalties, mimicry, and find important inputs. A common assumption underlying almost all work on understanding models, across these strategies, is that smaller models are easier to understand. Pazzani (2000, p. 10) points out that there "has been no study that shows that people find smaller models more comprehensible or that the size of a model is the only factor that affects its comprehensibility." Similarly, claims that symbolic representations like decision trees and rules are more comprehensible are based on researchers' intuitions without supporting studies in cognitive science. Pazzani argues that machine learning and data mining researchers need to pay more attention to cognitive factors that help humans understand and learn. Currently known factors for understanding new concepts include consistency with prior knowledge (Pazzani, 1991), defining concepts using the same predictors to facilitate contrasting different concepts (Billman & Davila, 1995; Billman, 1996),[4] and relationships between predictors and concepts that are consistent for all examples (vs. a subset, as in decision trees; (Pazzani, 2000)). The work in this chapter relies on the same smaller-is-better assumption for comprehensibility and is subject to the same criticism. Everything else being equal, we believe that smaller does imply easier to understand, but concede that other factors also contribute to understandability. More research is needed to operationalize and incorporate additional factors into learning understandable models.

The first strategy, *understandable by design*, designs the learning algorithm so that learned models will be understandable. Typically this involves using

[4]Pazzani (2000, p. 12) gives the example that "carnivores have sharp teeth and a short distance from eye to eye; herbivores have flat teeth and their eyes are further apart".

a model representation language that is easily understandable (e.g., small decision trees; algebraic expressions (Ferreira, 2001)), often with hard constraints on model complexity (e.g., small maximum tree size; limited length equations using only addition, subtraction, multiplication, and division). For example, Liu et al. (2000) propose an algorithm to re-represent a decision tree as a single general rule and a list of exceptions to the general rule. As a second example, Pazzani et al. (1997) add monotonicity constraints to a rule learning algorithm to prevent the algorithm from learning rules that violate prior domain knowledge. Modeling with Bayesian networks often uses an extreme version of this strategy in which the entire model structure is derived from prior domain knowledge. The understandable-by-design strategy works well when the restrictions and representation language are appropriate to the modeling task, but for complex domains like species distribution the limited modeling flexibility results in less accurate models. This can be seen in Table 5.1: the single CART tree (with 1,357 leaf nodes) performed worse than the single MML tree (with 22,013 leaf nodes), and all the single trees performed worse than the bagged tree ensemble.

The *complexity penalty* strategy imposes a soft constraint on how complex a model can be by augmenting a learning algorithm with a complexity penalty that encourages the algorithm to find a good tradeoff between accuracy and model complexity. For example, De Falco et al. (2005) add heuristic penalty terms to the fitness function of a genetic algorithm that searches for good classification rules; rules that test more predictors or that use less restrictive range tests are penalized more heavily. The minimum description length (MDL) principle is a more formal approach that states that the best model is the one which provides the shortest description of the data. Description length is measured as the number of bits needed to encode the model plus the number of bits needed

to encode the model's mistakes on the training data. The MDL principle has been applied, for example, to constrain decision trees (Wallace & Patrick, 1993) and prune classification rules (Karalič, 1996). The MDL principle is non-trivial to apply since one must specify how models are encoded and how to search through the space of possible models.

An easier penalty to apply is the L_1 penalty, a sparsity penalty that forces the weights on many predictors to be zero. The idea was first presented in the LASSO algorithm (Tibshirani, 1996b) for linear regression with hundreds or thousands of predictor variables. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ be the regression coefficients and $\boldsymbol{X} = (X_1, \ldots, X_p)^T$ be the predictors. Then the penalized linear regression finds $\boldsymbol{\beta}$ that minimizes

$$\sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} \beta_j x_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{5.1}$$

The tradeoff parameter λ controls how sparse the solution is; when λ is large, many of the coefficients are 0. The name comes from the fact that $\sum |\beta_j|$ is the L_1 norm of the weight vector.[5] Sparsity penalties based on the L_1 norm have since been used in many other algorithms, including the elastic net (Zou & Hastie, 2005) and the group LASSO (Yuan & Lin, 2006).

Unlike the first two strategies, the *mimicry* strategy begins by learning an unconstrained, fully complex model. A simpler, understandable model is then trained to mimic (or approximate) the predictions of the complex model. A key

[5] One way to understand why an L_1 penalty encourages sparsity and other norm penalties do not (e.g. sum of the squared coefficients) is to consider finding the solution by gradient descent. The partial derivative of equation 5.1 with respect to coefficient β_j is

$$\sum_{i=1}^{N} (-2x_i(y_i - \beta_j x_i)) + \lambda \mathbf{1}(\beta_j \neq 0)$$

where $\mathbf{1}()$ is an indicator function that is 1 if the argument is true and 0 otherwise. Focusing on the portion from the penalty term, we can see that any non-zero β_j value is penalized equally, regardless of β_j's magnitude.

advantage to this approach is that artificial data points can be generated and labeled by the complex model, allowing the simple model to be trained with a virtually infinite number of training examples. Researchers have used mimicry to approximate neural networks with a single decision tree (Craven & Shavlik, 1996) and ensembles with a single decision tree or set of classification rules (Breiman & Shang, 1996; Domingos, 1998). The general consensus among these studies is that the mimic model performs better than a simple model trained directly from the original training data but not quite as well as the complex model. The mimic's fidelity to the complex model, accuracy on test data, and simplicity depend heavily on (a) the domain, (b) how artificial data is generated (Breiman & Shang, 1996; Domingos, 1998; Buciluǎ *et al.*, 2006), and (c) how much artificial data is used for mimic training. It would be interesting to apply mimicry to the species distribution models below to see if a mimic retains enough accuracy to be worth studying.

The research in this chapter uses the last strategy: *find important predictors* that strongly affect the model's predictions. This strategy is less ambitious than the others, but when combined with methods for visualizing the relationships between inputs and outputs (§ 5.5) also yields insight into how a complex model makes predictions. Besides sensitivity analysis (described in § 5.1) and the tree statistics we present below, this category also includes the numerous methods for variable selection (briefly reviewed in § 2.7), importance measures derived from surrogate splits in decision trees (see below), and partial derivative analysis of weights in neural networks (Intrator & Intrator, 2001).

Breiman *et al.* (1984, pp. 140–142, 146–150) proposed using surrogate splits to measure the importance of predictor variables used in decision tree models. Let X_m be the predictor used to split training data at some node in the tree. A **surro-**

gate split is an alternative split for the node using a different predictor, say X_a, that splits the training data into similar subsets. In other words, the surrogate split approximately agrees with the main split that is actually part of the tree. Surrogate splits are one mechanism for handling missing values when making predictions (i.e., if the value for X_m is missing, use the surrogate split on X_a instead). Breiman et al. noted that predictor importances could be estimated by how much the predictor reduces impurity (uncertainty) when it is used in internal tree nodes (similar to the tree statistics in § 5.3). Unfortunately, this measure is unreliable because only the best predictor test is installed during tree growing; there may be other predictors that would have been almost as informative. The impact from this masking effect is most serious for small trees (e.g., CART-style trees) because there are only a few chances (internal nodes) for a predictor to demonstrate its relevance. To obtain reliable importance measures for all predictors, Breiman et al. use surrogate splits to estimate importance. Specifically, for every node in the learned tree, the best surrogate split for predictor X_a is found; X_a's importance is the sum of information gained from all the surrogate and installed splits on X_a. This process is repeated for each predictor to generate a ranking of predictors by importance. Measuring importance via surrogate splits requires additional computation to identify the surrogates at each internal tree node after the tree is learned, and this computation can be substantial when the tree is large and there are many predictors.[6] Our work has not explored surrogate splits, both because of this extra cost and because of the fact that surrogate splits are not implemented in most decision tree packages (i.e., this feature is not readily available to practitioners and analysts).

Our work is similar to embedded feature selection techniques that select the

[6]The computation per tree node is $O(np)$, where n is the number of examples and p is the number of predictors.

most important predictors by training a machine learning model and studying the internal model weights. For example, recursive feature elimination removes predictors with small influence on the decision boundary of a support vector machine (Guyon *et al.*, 2002). Contemporaneously with our work, Tuv et al. (2006) independently proposed measuring the importance of predictors in tree ensembles based on tree statistics. While our primary goal is understanding the ensemble, they use the importance measures for variable selection. Their work uses random forests instead of bagged decision trees, which allows them to efficiently add extra probe predictors that are pure noise. The probes are ranked by importance along with the other predictors, and predictors ranked below the probes are discarded. In other words, the probes define a cutoff in the importance rankings for predictors that can be discarded. Followup work incorporates surrogate splits to identify and remove additional predictors that contain redundant information (Tuv *et al.*, 2009).

5.3 Fast Tree Statistics for Measuring Predictor Importance

This section proposes several tree-based statistics for measuring predictor importance. All of the methods leverage the fact that we are using ensembles of decision trees. We can inspect the learned trees to see which predictors have been selected to subdivide the training data. Because selected predictors separate positive and negative observations, they are clearly important predictors. If a predictor is "important" for many of the trees in the ensemble, then we have strong evidence of its overall importance. The main challenge is defining a good measure to quantify a predictor's importance in a tree and in an ensemble

of bagged trees.[7]

We have implemented several ranking methods that use only the information about the tree structure and how a training set is partitioned by the different trees. This information is available once the ensemble is built so there is no need to generate new models or new predictions in order to calculate these rankings. This is a clear advantage over black box methods like sensitivity analysis or variable selection. Each statistic can be computed for all predictors by traversing all the trees in the ensemble, and computational complexity is $O(m)$ where m is the total number of nodes in the ensemble.[8] For an ensemble model learned from the PFW data described in section 5.4.1, we can compute the complete ranking of *all* predictor variables in less than 2 minutes (no matter which of the methods introduced below we are using). Compared to 4–5 minutes *per predictor* for sensitivity analysis, this is a factor of 500 speedup!

The importance score of a predictor for the tree ensemble is computed by summing the importance scores on the individual trees. To illustrate the differences between the methods, we will use the simple tree shown in Figure 5.1. It splits on three attributes: A, B, and C. The training set has 100 examples; numbers in parentheses indicate the number of examples affected by the corre-

[7]The tree statistics are also applicable to other kinds of decision tree ensembles such as random forests (Breiman, 2001a) and boosting (Freund & Schapire, 1996). To use with boosting, the statistics should be adapted to incorporate the weights boosting places on each training example, and the contribution from each base model in the boosted ensemble should be proportionate to its weight in the ensemble.

[8]Comparing this complexity to the $\Theta(npt)$ complexity of sensitivity analysis is tricky because the number of nodes in a decision tree depends heavily on several factors including a) how aggressively the tree is pruned, b) the noisiness of the data, c) the number of continuous predictors, and d) the size of the training set. The last factor is potentially worrisome: Oates and Jensen (1997) empirically observed that the number of nodes in a decision tree grows linearly with the size of the training set, and training set sizes are large for many interesting application domains. On the other hand, sensitivity analysis depends on test set size n which generally is of the same order as the training set size. The comparison is further complicated by the facts that a) m can depend on the number of predictors p, and b) prediction time t depends on m (but probably sub-linearly, e.g., logarithmically). In practice we expect m will be significantly smaller than the product npt, and empirical timing comparisons for the PFW data bear this out.

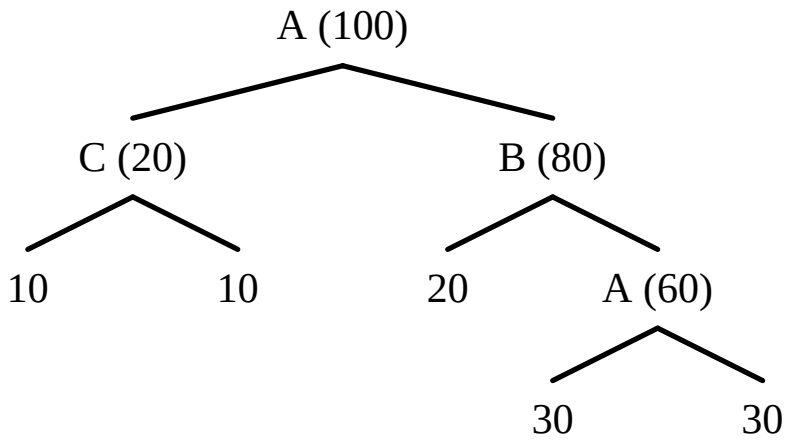

Figure 5.1: Sample decision tree.

sponding split (i.e., the number of examples in the corresponding subtree). We consider the following methods for computing predictor importance scores on a single tree.

Number of nodes (#nodes):

A predictor's score is the number of nodes in the tree that selected the predictor for the split. In our example predictor A gets importance score 2, while B and C receive importance scores of 1 each. This method will give too much weight to continuous predictors, because the tree can split on them more often. The other methods address this issue.

Weighting by height (height):

Greedy tree growing algorithms usually choose the most important predictors early, so they appear higher in the tree structure. This method weights each node inversely proportionally to the length of the path from it to the root. The root itself is considered to have importance 1, so in the example predictor A receives importance $1 + 1/3$ (importance of root + importance of the rightmost subtree split), while predictors B and C each have importance $1/2$. The example in Figure 5.1 illustrates a prob-

lem with height-based weighting. Predictors B and C receive the same weight, whereas splitting on them affects different numbers of cases in the data set. To correct for this, the following methods take into consideration the number of training cases affected by the split.

Weighting by size of training set — multiple counting (multiple):

This method weights a node by the number of training cases in its subtree, i.e., the cases affected by the split at this node. In the example, predictors A, B, and C receive scores of 160, 80 and 20, respectively.

Weighting by size of training set — single counting (single):

As with *#node* ranking, there is a risk that continuous predictors will be over-weighted when using the *multiple* counting of training points. In the example, the 60 records in the lower-right subtree with parent node A are counted twice towards A's score. To fix this problem, *single* counting assigns weight zero to all nodes that have an ancestor with the same split predictor. In the example A receives an importance score of 100 instead of 160, while the scores for B and C do not change.

Weighting by size of training set — giving weight to the path (path):

This method compromises between *single* and *multiple* counting. Intuitively, training records from every leaf are distributed evenly between the splits on the path from the root to the leaf. Each split is still counted, even if there is another split on the same predictor in an ancestor node. In our example, the 30 records from the rightmost leaf are distributed between the two splits on A and the one split on B, i.e., 20 points go to A and 10 to B. Similarly, the 10 points from the leftmost leaf are given to A and C, in this case 5 points to each. Counting from left to right, A receives an importance score of $5 + 5 + 10 + 20 + 20 = 60$, B gets $0 + 0 + 10 + 10 + 10 = 30$

and C gets $5 + 5 + 0 + 0 + 0 = 10$. It is worth mentioning that importance scores for all predictors sum to the size of the training set used to build the tree. A similar method was used by Friedman and Popescu (2008) for estimating predictor importance in an ensemble of rules.

The next section examines and compares these importance measures with sensitivity analysis on models of avian species distribution.

5.4 Empirical Comparison of Importance Measures

The efficiency of the tree statistics defined above is only useful if the statistics score important predictors above unimportant predictors. This section empirically evaluates the statistics in two ways. First, we use the statistics and sensitivity analysis to rank the predictors used in models of species distribution (§ 5.4.3) and compare how correlated the importance measures are to each other. Second, we measure the performance of new models that are trained using only the most highly ranked predictors from each of the importance measures (§ 5.4.4).

We evaluate the statistics in the problem domain that motivated them: modeling the wintertime distribution of bird species in North America. Before presenting the results of the comparisons, we describe the PFW data set used for training the distribution models (§ 5.4.1) and summarize how the models were built (§ 5.4.2).

5.4.1 Description of the PFW Data

The data examined come from Project FeederWatch (PFW, `http://birds.cornell.edu/pfw`), a winter-long survey of North American birds observed at bird feeders. PFW has been running since the winter of 1987-88, and as of

March 2005 had over 1 million submissions, which report a total of 11.7 million bird sightings. Participants report *all* observed species; therefore the data also imply the absence of all species that were not reported. For the 100 most interesting species, this adds about 90 million "bird absence" records to the data set.

Each PFW location and submission is described by multiple predictor variables which are provided by project participants. These predictors can be roughly grouped into variables related to observer effort, weather during the observation period, and attractiveness of the location and neighborhood area for birds. The observer-provided predictors were supplemented with several hundred additional descriptions of the environment that came from a variety of geographic data sets, e.g., the U.S. Census Bureau's 2000 census (human impact), the USGS National Elevation Dataset, the USGS National Landcover Dataset, and various descriptions of local climatic conditions (e.g., monthly snow depths, wind speed, temperature) from the National Climatic Data Center's Climate Atlas of the United States.

After data were screened to exclude those observation records that were improperly submitted, or did not have sufficient fields to be included in the analysis, a total of about 800,000 observation records with 197 potentially important predictors were available for analysis. Recall that each record indicates the presence of a set of species and implies the absence of all other species. Hence for each bird species there are 800,000 records about its presence or absence, constituting a massive amount of information about bird occurrence. These records still included a considerable fraction of fields where the participant-reported variables were missing.

Because of the ecologists' expectation that species' population trends and

the most important influences on any species' distribution will vary across the continent, we subdivided the continent into ecologically-relevant units, using 37 existing Bird Conservation Regions (BCRs; see `http://www.nabci-us.org/map.html`). There are 600+ BCR-species pairs with sufficient data for machine learning analysis. In this evaluation we limited our attention to the simpler, but nevertheless challenging, problem of analyzing data from nine BCR-species pairs; specifically we analyzed the american goldfinch, dark-eyed junco, and house finch in BCRs 5, 22, and 30. The high-level conclusions from these analyses are similar so we only present results for the house finch model in BCR 30 (the U.S. Atlantic coastal plain region from southern-most Maine to northern-most Virginia (U.S. NABCI Committee, 2000)). BCR 30 has 92,514 observations; of these, the house finch is present 55,860 times.

5.4.2 Modeling Details

We trained bagged decision tree models for each of the nine BCR-species pairs listed in the previous section. All ensembles consisted of 100 ID3 trees built using the IND package (Buntine & Caruana, 1991). Experiments with other tree types indicated that the choice of tree type had fairly little effect on the ensemble performance. The data sets were partitioned into roughly 2/3 training and 1/3 testing. Learning each ensemble on a single processor PC (3.6 GHz, 1 GB RAM) took about 2 hours.

For our sensitivity analysis we selected a diverse set of commonly used performance measures: accuracy (ACC), root mean squared error (RMS), and area under the ROC curve (AUC). (See section 2.3 for a review of these performance measures.) All performance numbers were measured on the test set.

5.4.3 Correlations Between Importance Measures

Several patterns emerged from comparing how the different importance measures ranked predictors. First, sensitivity analysis produced different rankings depending on the performance metric used to measure the loss from perturbing each predictor. The rankings from sensitivity analysis using RMS and AUC as the performance measures were highly correlated for the most important predictors (ranks 1–20) and moderately correlated for the top 100 predictors; they were only loosely correlated for predictors ranked below the top 100 (Figure 5.2a). In contrast, the ranking from sensitivity analysis using ACC as the performance measure was almost entirely different from the other sensitivity analysis rankings and only agreed with the others for the top 10 or 15 predictors (Figure 5.2b). Because accuracy is known to be a high variance measure, while RMS is stable, we have more confidence in the results of RMS sensitivity and AUC sensitivity. Consequently, we used the RMS sensitivity ranking for assessing the quality of the tree statistics.

Second, the tree statistics formed two groups of importance measures that were intra-correlated. All of the statistics that relied on counts of data points—i.e., *multiple*, *single*, *path*—produced importance rankings that were almost perfectly correlated (Figure 5.2c). The only differences were localized reorderings of predictors. This result is surprising, because different ways of handling continuous attributes could in theory have significant influence on the resulting rankings. For this data set at least, multiply counting training points does not seem to be a concern. The *height* and #*nodes* statistics similarly produced nearly identical rankings (graph not shown).

Third, the *height* importance measure—and by extension, the #*nodes* ranking—did not correlate at all with sensitivity analysis (Figure 5.2d). It seems that

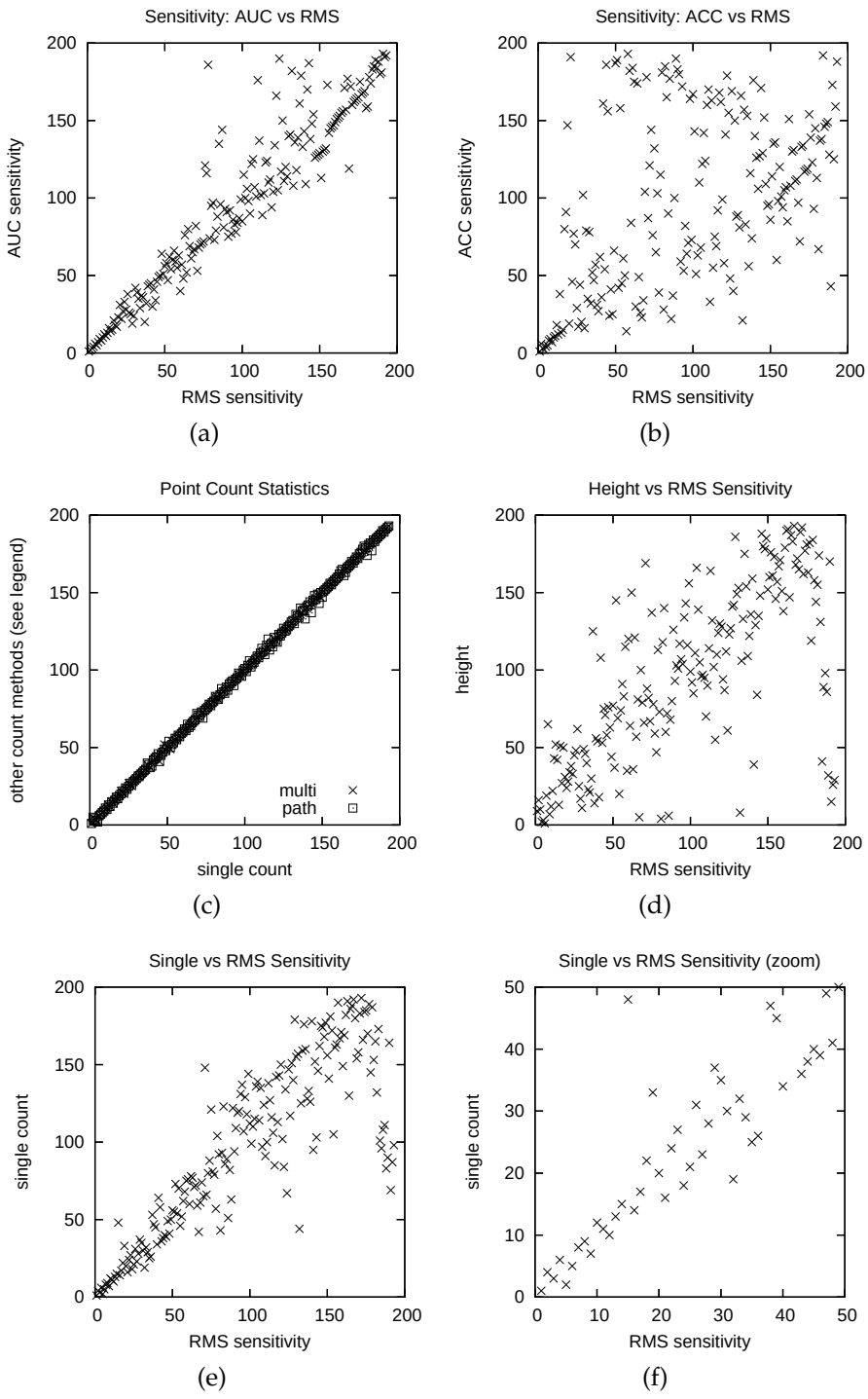

Figure 5.2: Correlations between different importance ranking measures. Each graph plots the rank positions of the predictors, according to two different importance measures. Ranks are ordered by decreasing importance; i.e., an importance measure assigns rank 1 to the predictor with the highest importance score. Points (i.e., predictors) close to diagonal line are ranked similarly by both measures.

predictors close to the root of the trees are not necessarily the most important predictors. Subsequent tests showed that results of these methods are less reliable than those of *single*, *multiple* and *path* (see Section 5.4.4).

Fourth, the point count statistics (*single*, *multiple*, and *path*) correlated well with sensitivity analysis for the top 20 or 30 predictors but became less correlated for less important predictors (Figure 5.2e; see Figure 5.2f for close up of most important predictors).

The most important result here is the fact that the fast point count statistics essentially identify the same top 20 predictors as the more expensive sensitivity analysis. This result is also true for the other 8 BCR-species pairs we analyzed. As long as most of the model's performance comes from these top predictors, we can take advantage of the faster methods without sacrificing result quality. The next section measures the performance of models trained using only these important predictors.

5.4.4 Sanity Check

There is no guarantee that taking the top-ranked predictors from any of these importance measures will yield an ensemble with good predictive power. While prediction accuracy is not the only goal of this study, it is a necessary precondition. Clearly we cannot hope to learn something about this domain by studying inaccurate models. Also, ecologists are interested in comparing the important predictors of a species' occurrence in different BCRs. This can be achieved by comparing rankings, but only after checking that some minimum predictive performance is met in all analyses.

As a sanity check, we compared the performance of bagged trees trained using all predictors with bagged trees trained using only the top 20 predictors

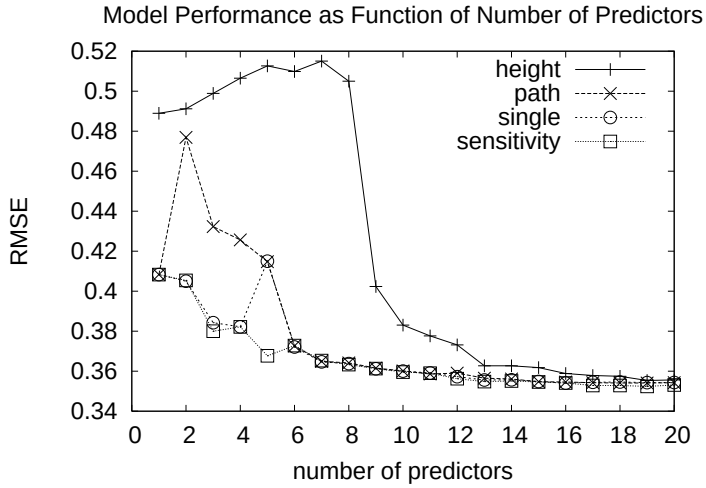

Figure 5.3: Performance as a function of the number of predictors used for training. Each line represents a different method for ordering predictors by importance—yielding slightly different sets of predictors.

from the different importance rankings. With all predictors, the bagged trees achieve a RMS of 0.3469, accuracy of 0.8336, and AUC of 0.9012.

Figure 5.3 plots the ensemble's RMS performance when only the top N predictors from each ranking are used, for different values of N. Because the rankings differ from each other, different predictors are included at each point for the different lines (see Table 5.2 for predictors included at each step). The overall pattern is similar for accuracy and ROC area, so we omit those graphs.

We make several observations from Figure 5.3. First, the ensembles built using only 20 predictors perform quite well, although not quite as well as ensembles using all the predictors. The top 20 predictors do seem to catch most of the predictive power found in the full predictor set. This gives us some confidence in relying on these measures as indicators of which predictors are important for modeling the PFW domain.

Second, while the rankings from single counting, path counting, and RMS sensitivity analysis show similar behavior, the height-based ranking behaves

Table 5.2: Top-20 predictors from four predictor importance measures; 'num-feeders_' is abbreviated as 'nf_'.

rank	RMS sensitivity	height	path	single
1	latitude	dayselapsed	latitude	latitude
2	longitude	yearseason	dayselapsed	halfdays
3	nf_hanging	halfdays	nf_hanging	nf_hanging
4	halfdays	temp_lo_atleast	longitude	longitude
5	yearseason	temp_hi_atleast	halfdays	dayselapsed
6	dayselapsed	precip_len_atleast	yearseason	yearseason
7	nf_thistle	effort_hrs_atleast	nf_thistle	nf_thistle
8	ave_fam_sz	snow_dep_atleast	effort_hrs_atleast	effort_hrs_atleast
9	effort_hrs_atleast	latitude	ave_fam_sz	ave_fam_sz
10	asian	nf_hanging	elev_ned	elev_ned
11	elev_ned	nf_ground	asian	asian
12	evgr_trees_atleast	nf_suet	pop00_sqmi	nf_suet
13	nf_suet	longitude	nf_suet	count_area_size
14	gcsnow2912	snow_cov_atleast	vacant	pop00_sqmi
15	pop00_sqmi	nf_platfrm	count_area_size	vacant
16	vacant	pop00_sqmi	elev_gt30	black
17	count_area_size	elev_gt30	black	age_65_up
18	other	nf_water	ave_hh_sz	elev_gt30
19	elev_gt30	asian	age_65_up	ave_hh_sz
20	ave_hh_sz	black	houden	houden

very differently. This agrees with the finding above that the height importance measure is not as highly correlated with the other measures.

One surprising aspect of this graph is that all the lines go up at least once: *path* at predictor 2, *single* at predictor 5, *sensitivity* at predictor 6, and *height* for the first half of the graph. This phenomenon is partly caused by the predictor `dayselapsed`; whenever it is added, performance gets worse in this graph. Given that all the measures rank this predictor highly, and the ecologists believe it to be an important predictor, this is rather surprising.

Overall, we have identified methods for analyzing the ensemble model that produce very similar rankings of predictor importance. We have also shown that the resulting rankings are reasonable: models generated using only the top

20 predictors show good performance.

Once important predictors are identified, the next step to understanding a model is visualizing the relationships between important predictors and the model's outputs. The next section describes how we visualized the effect of the important predictors found above and presents some sample results for important predictors.

5.5 Visualizing Important Predictors

As described in section 5.1, one of the main goals for this project is identifying predictors that are important in predicting the distribution of bird species. Section 5.3 presented several heuristic methods for finding potentially important predictors. In order to decide if a certain predictor requires closer examination, ecologists need information about how the predictor affects the probability of observing the bird.

To provide such information we estimate and plot the probability of spotting the bird given different values of the predictor in question. Figure 5.4 contains several examples of this kind of graph; for convenience we will refer to these graphs as *trend plots*. The rest of this section describes how we generate trend plots and discusses some sample plots.

5.5.1 Generating Trend Plots

We explore two methods for plotting trends: 1) computing conditional probabilities directly from the data; and 2) computing Friedman's *partial dependence function* (Friedman, 2001) for the predictor of interest, using the previously learned model to estimate probabilities. We will refer to these methods as *data* and *par-*

tial, respectively.

Data: Given each value of the predictor of interest, we compute the probability of seeing a bird after marginalizing out the other predictors. This is just the mean of *all* the points in our data set that have the given value of that predictor. Points lacking a value for the predictor (i.e., missing value) are not used. Continuous predictors are discretized into 5% quantiles to yield twenty distinct values for plotting, with each data point summarizing roughly the same number of data records. The top of each bin (quantile) is plotted on the x-axis. Note that continuous predictors are discretized identically for both methods.

Partial: For each value v to plot for predictor X, create an artificial data set D_v by setting $X = v$ for *all* the points in the test set.[9] Each artificial data set is labeled by the (previously learned) bagged tree model. The probability of observation when $X = v$ is computed by averaging the predictions for the set D_v. Missing values are a non-issue with this method.

The motivation behind partial dependence functions is that the target predictor X may have high correlation with another predictor Y for *some* values of X. If X is not an important influence but Y is, marginalizing to find X's influence on seeing a bird (using the data method above) can make X look like an important indicator for values where it correlates well with Y (the truly important variable). As a result, perceived observation trends as a function of X may be exaggerated or may not exist at all.

Substituting $X = v$ for all points breaks up potential covariances and forces the model to focus more on the impact of X having value v. The only thing that changes between plot points is the value of X — holding all other predictors

[9]The mean value of each quantile is used as the substitution value for continuous predictors.

constant in some sense, while still maintaining the natural distribution of their values.

In theory, partial dependence functions can produce misleading plots in cases where we generate many new points in regions of the predictor space unsupported by our data. The model, which was not trained on the data from those regions, can produce unpredictable results that will harm our trend plots. The detailed description of a similar problem can be found in (Hooker, 2007). In our analysis, we have not discovered this problem yet.

5.5.2 Sample Trend Plots

Because computing partial dependence functions for all 197 predictors is too computationally expensive, we examined the top 20 predictors from the single counting ranking (see Section 5.3). Figure 5.4 shows six trend plots. Brief descriptions of the predictors plotted are given below.

Each graph shows the probability of observing the house finch in BCR 30 as a function of a given predictor. As a general rule, the *partial* plots are much smoother than the *data* plots, which exhibit much more local variance. In most cases, however, both methods show the same general trends. Most of the comments below will focus on the *partial* plots because they are easier to read and interpret.

`yearseason`: The observed decline in occurrence is consistent with ecologists' background knowledge that a novel bacterial pathogen, first appearing in 1994, has caused declines in abundance of house finches across northeastern North America.

`latitude/longitude`: As we know from Table 5.2 these two predictors both have high importance ranking. The analysis in section 5.6.1 below also

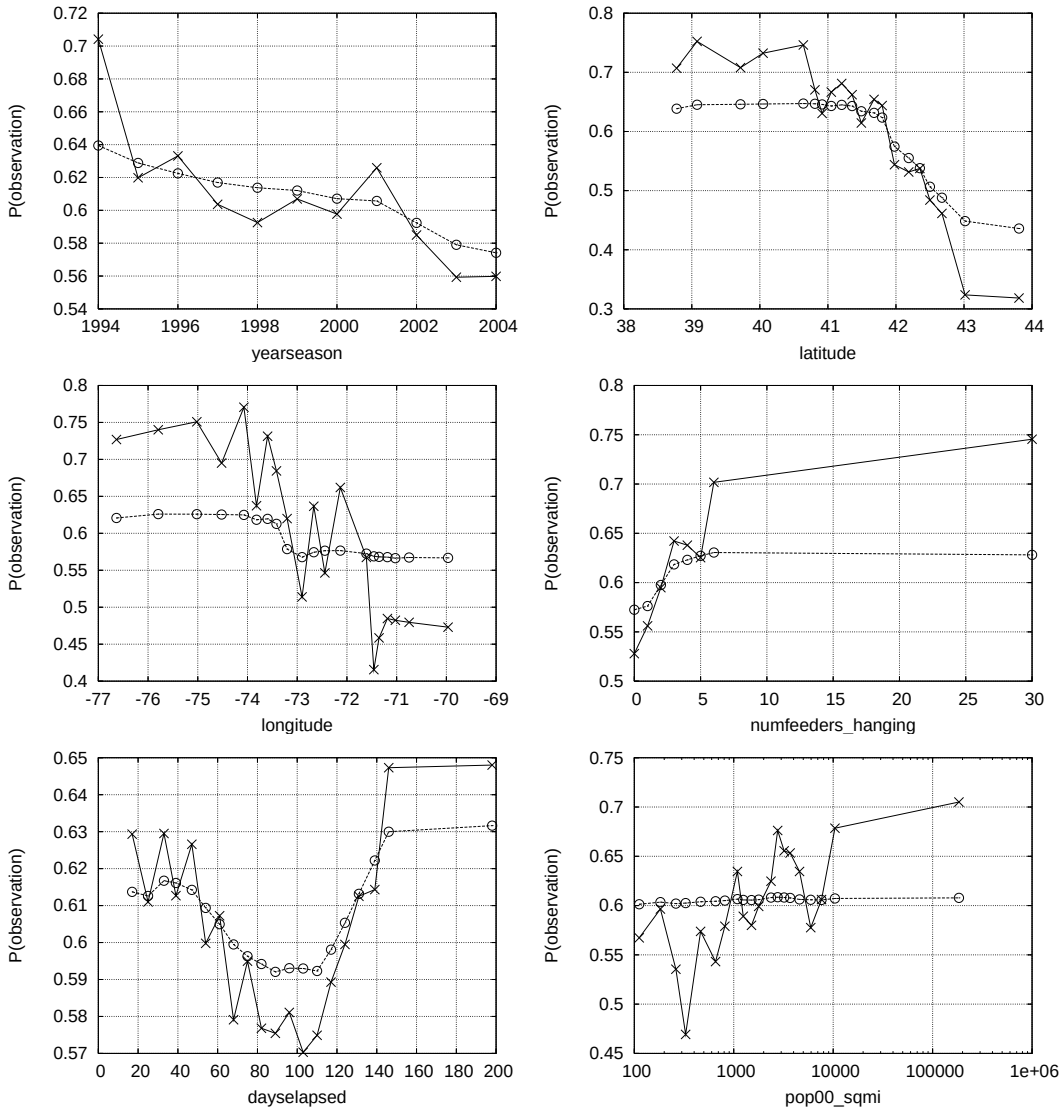

Figure 5.4: House finch observation trends in BCR 30. Each graph shows the *data* (x line) and *partial* (o line) plots for a different predictor. Note that the y-axes cover different ranges.

shows that they are highly associated with each other and with many other predictors. We believe that these variables describe spatial gradients and possibly act as proxies for other attributes that also exhibit spatial variation. The greater range of variation in the latitude effect may be due to the large North-South orientation of BCR 30.

`numfeeders_hanging`: This predictor counts the number of hanging bird feeders in the observation area. As the number of feeders increases from 0 to 5, we see an increasing probability of observation. The plateau effect past 6 feeders suggests that once there are sufficient feeders adding more does not increase the chances of seeing a bird.

`dayselapsed`: This variable counts the number of days elapsed since the beginning of the PFW season. Since the season begins on November 1, day 31 is the beginning of December (for example). The observed pattern of probability of occurrence is consistent with the known partial-winter migratory behavior of house finch populations in the eastern United States, where a proportion of the winter population migrates.

`pop00_sqmi`: This is the *human* population per square mile, as measured during the 2000 census. This is a good example of the *partial* plot differing from the *data* plot. The former suggests that the influence of population density on house finch occurrence is relatively small, despite the fact that the model considers it important. The latter, however, would indicate that the probability of seeing a house finch increases dramatically as population density goes up. Taken together, it seems more likely that population density correlates with other important indicators (especially given the large peaks and valleys in the data line).

The example of `pop00_sqmi` also shows that a predictor can be important

for model prediction even though its *partial* line is close to a flat line. A flat trend line does not prove that a predictor is unimportant. Rather, it just shows that in this marginalized setting the predictor does not carry much predictive weight. Combined with other predictors, however, it may be very important for making good predictions. Therefore, examining trend plots is not a viable way to identify important predictor sets by itself.

5.6 Complications from Correlated Predictors

Interpreting variable importance rankings and the effects of individual predictors is relatively easy when predictors are uncorrelated. Unfortunately, when dealing with environmental data, variables tend to be correlated. This is especially true as the number of predictors increases. For example, weather, climate, elevation and landcover predictors commonly exhibit complex patterns of association. When they exist, inter-correlations affect the interpretation of the relative importance of selected predictors as well as the effects of those predictors.

Correlated predictors complicate finding the important predictors in two ways. First, correlation causes **masking effects** during model learning. Let A and B be two informative predictors that are highly correlated. If A is slightly more informative than B then a decision tree learning algorithm will tend to use A in the learned model and ignore B completely. Any importance measures based on this model—be it tree statistics, sensitivity analysis, or variable selection—will show that A is important and B is useless. Alternatively, correlation can cause **dilution effects**, also during model learning. If predictors A and B contain the same information or almost the same information then the choice of which to include in the model is essentially random. The different models in an ensemble will pick one of them at random, and the importance of both pre-

dictors will be cut in half. Dilution effects also arise within a single decision tree when the predictors are continuous: the tree can split on a continuous predictor multiple times, and each of these splits randomly chooses A or B.

In summary, the importance measures studied above can be used to choose reasonable small sets of predictors, but they will not find many other sets of important predictors that perform as well or even better. Ecologists are interested in these alternate predictor sets because some predictors are more biologically relevant. For example, average snowfall is correlated with latitude. Birds probably do not choose their wintering grounds because they like the latitude; rather, they choose wintering grounds based on temperature and the availability of food and water. Latitude is only an indirect measure of a process that influences species occurrence. Given alternate choices for important predictors, ecologists could choose the predictor set they felt was most biologically relevant.

The rest of this section explores one way to find alternate predictor sets: expanding the set of important predictors by finding which other predictors are highly associated.

5.6.1 Alternate Predictor Sets

Ecologists would like to have tools to explore how inter-correlations among important predictors affect predictive performance and interpretation. As a first step towards addressing this problem we analyzed the effects of swapping pairs of strongly related predictors on the predictive performance of our models.

In order to identify pairs of related predictors we measured the association between variables Y and X as the percent of variation explained from predicting the rank of Y's values based on the rank of X's values and the square of the rank

of X's values. Conceptually, the model is:

$$\text{rank}(y) = \beta_1 \text{rank}(x) + \beta_2 (\text{rank}(x))^2 \tag{5.2}$$

This squared rank correlation measure is a simple generalization of Spearman's rank correlation (Harrell, 2001). It can detect nonlinear and non-monotonic relationships between X and Y. This measure is applied to categorical predictors X by representing X as a series of indicator variables for each category. Missing values are handled by deleting pairs of values, rather than deleting all rows of X having any missing variables.

We computed the predictor associations between each of the top 20 predictors, as determined by the single count method, with all other 196 predictors. In Table 5.3 we report the 5 most strongly associated predictors for each of the top 20, with the squared rank correlation noted in parentheses. The squared rank correlation ranges from 0 to 1, with larger values indicating stronger associations.

5.6.2 Exploring Alternate Models by Swapping Predictors

Using the pairwise associations in Table 5.3, the ecologists selected predictor swaps for further exploration. For each swap experiment, a model is fit with the top 16 predictors as determined by the single-counting method (§ 5.3) with one (or two) variables changed. We report the performance of a bagged tree model containing 100 ID3 trees. The model is trained on the full data set and performance is evaluated by using out-of-bag estimation (Bylander, 2002).

Table 5.4 lists the resulting performance of several swaps. The first column contains the predictors involved in the swap, with the arrow indicating which predictor was replaced. For example, in experiment four we swapped annual

Table 5.3: Top predictor associations. The 20 most important predictors (determined by single count method) are shown along with the five predictors most strongly associated with them. Highly associated predictors (> 0.8 association) are italicized.

Predictor	**Top 5 Associated Predictors**
latitude	*gcsnow1413 (0.876), gcsnow1412 (0.860), longitude (0.844), gcsnow1403 (0.831), gcsnow1401 (0.829)*
halfdays	day2_pm (0.550), day1_pm (0.482), day2_am (0.267), day1_am (0.134), effort_hrs_atleast (0.094)
numfeeders_hanging	numfeeders_suet (0.161), numfeeders_thistle (0.135), numfeeders_ground (0.063), numfeeders_water (0.055), numfeeders_fruit (0.047)
longitude	*latitude (0.841), gcslvp6101 (0.837), gcslvp6112 (0.822), gcslvp6111 (0.804)*, gcslvp6102 (0.794)
dayselapsed	snow_dep_atleast (0.025), snow_cov_atleast (0.013), snow_crusty (0.009), fed_in_nov (0.003), temp_lo_atleast (0.001)
yearseason	snow_cov_atleast (0.354), precip_len_atleast (0.113), hab_desert_scrub (0.063), yard_type_woods (0.057), hab_orchard (0.054)
numfeeders_thistle	numfeeders_suet (0.146), numfeeders_hanging (0.132), fed_in_aug (0.082), fed_in_jul (0.081), fed_in_jun (0.072)
effort_hrs_atleast	halfdays (0.102), day1_pm (0.056), day2_pm (0.052), day2_am (0.029), count_area_size (0.015)
ave_fam_sz	ave_hh_sz (0.791), med_age_f (0.281), med_age (0.273), age_5_17 (0.244), med_age_m (0.237)
elev_ned	*elev_gt30 (0.932)*, gcsnow3513 (0.342), gcsnow2901 (0.313), gctemp0312 (0.300), gcsnow2902 (0.271)
asian	hispanic (0.285), mult_race (0.225), females (0.175), males (0.166), age_22_29 (0.161)
numfeeders_suet	numfeeders_hanging (0.164), numfeeders_thistle (0.147), numfeeders_ground (0.081), numfeeders_platfrm (0.078), numfeeders_water (0.060)
count_area_size	dcid_trees_atleast (0.067), fru_trees_atleast (0.066), numfeeders_ground (0.064), dcid_shrbs_atleast (0.061), dcid_any_atleast (0.059)
pop00_sqmi	*houden (0.943)*, housing_density (0.370), lu_nlcd (0.345), hispanic (0.229), other (0.192)
vacant	hsehld_1_m (0.369), hsehld_1_f (0.289), renter_occ (0.237), mhh_child (0.212), households (0.202)
black	mult_race (0.370), other (0.343), hispanic (0.338), longitude (0.263), age_22_29 (0.257)
age_65_up	hsehld_1_f (0.438), marhh_no_c (0.415), households (0.387), age_50_64 (0.335), females (0.310)
elev_gt30	*elev_ned (0.931)*, gcsnow3513 (0.353), gcsnow2901 (0.333), gctemp0312 (0.312), gcsnow1412 (0.278)
ave_hh_sz	ave_fam_sz (0.790), hsehld_1_f (0.440), hsehld_1_m (0.315), renter_occ (0.310), marhh_chd (0.287)
houden	*pop00_sqmi (0.943)*, housing_density (0.349), lu_nlcd (0.343), renter_occ (0.207), hispanic (0.190)

Table 5.4: Results for swapping predictors.

	ACC	RMS	AUC
starting performance	0.8255	0.3546	0.8929
pop00_sqmi ← houden	0.8252	0.3550	0.8924
elev_ned ← elev_gt30	0.8264	0.3543	0.8932
ave_fam_sz ← ave_hh_sz	0.8252	0.3543	0.8933
latitude ← gcsnow1413	0.8258	0.3539	0.8937
longitude ← gcslvp6113	0.8244	0.3550	0.8926
latitude ← gcsnow1413, longitude ← gcslvp6113	0.8255	0.3547	0.8929
remove latitude and longitude	0.8257	0.3551	0.8925

average snowfall (`gcsnow1413`) for `latitude` with the goal of discovering the degree to which latitude's predictive power can be explained by annual average snowfall, given that the other top 15 attributes are included in the model. The top row shows "starting performance" — performance of the model built using only the top 16 predictors according to the single-counting method. The following rows show results of different swaps for one or two of the top 16 predictors.

In the first experiment we swapped housing density (`houden`) for population density (`pop00_sqmi`), two census variables with nearly identical definitions and a very strong pairwise association. In the second swap we exchanged two elevation measures; `elev_ned` has a resolution of about 30m by 30m and `elev_gt30` has a resolution of about 1km by 1km. In experiment three we swapped average household size(`ave_hh_sz`) for average family size (`ave_fam_sz`). In the fourth experiment we tested the degree of association between latitude and level of snowfall(`gcsnow1413`). In experiment five we swapped average annual precipitation (`gcslvp6113`) for longitude. Experiment six applied the swaps of both experiments 4 and 5. In the final experiment, we completely removed `latitude` and `longitude`, without replacing it by any other predictor.

The small change in performance for the first three swaps is not surprising given the similarities in the predictor definitions and their strong pairwise associations. The ability to change snowfall for latitude and precipitation for longitude, without affecting overall performance much, is somewhat more surprising, because in the sensitivity analysis adding noise to latitude or longitude resulted in the largest loss of performance, whereas `gcsnow1413` and `gcslvp6113` were ranked only 106 and 135. The low ranks of `gcsnow1413` and `gcslvp6113`, despite their strong associations with highly ranked variables, are likely due to masking effects.

An additional explanation is that latitude and longitude are highly associated with each other in this data set [10]; thus, any information "lost" by swapping `gcsnow1413` for latitude can be extracted from the longitude variable. Moreover, latitude and longitude may each be associated with some of the other 14 or 15 predictors used in these models, many of which have non-uniform spatial distributions. This would explain the second-to-last row in Table 5.4 where both latitude and longitude can be swapped with no substantial changes in performance. If we go further and remove latitude and longitude from the set of predictors *without replacing them*, performance continues to remain unchanged.

To summarize these results, swapping predictors may be useful for exploring alternative predictors, but to be effective the initial set of predictors should be minimal. In other words, removing any predictor from the initial set should hurt model performance. One way to ensure this is to find a small set of important predictors (perhaps using the importance measures from section 5.3) and then apply backwards variable elimination to remove unnecessary predictors in a stepwise fashion. Once all variables in the important set are necessary for good predictions, swapping predictors should be more informative.

[10] We believe this to be a side effect of the shape and orientation of BCR 30.

5.7 Conclusions

Finding important predictors for predicting the presence or absence of species is a major goal of ecology. The large data size, large number of predictors, and inherent quality issues of data collected by citizen science projects make this a challenging problem. In this chapter we studied techniques that determine the importance of a predictor by how heavily an accurate data mining model relies on the predictor for its predictions. Expensive approaches like variable selection did not scale, resulting in poor response times even for this limited study.

We presented very fast heuristics for measuring predictor importance that are based on analyzing the structure of decision trees. An interesting outcome of this study is that all heuristics that measure importance by the number of training cases affected by a node split produce almost identical predictor rankings. Furthermore, the top 20 of these rankings are also highly correlated with those computed by more expensive sensitivity analysis.

We showed how to extend the initial set of important predictors using predictor-association analysis. This enables ecologists to explore alternative sets of important inputs by replacing some top-ranked predictors with highly associated alternates. Once a small set of interesting predictors is identified, expensive trend plots can be generated to gain a better understanding of how certain predictors affect the observation probability for a species.

The analysis presented in this chapter was applied to 9 BCR-species pairs, for a single project (PFW). As pointed out earlier, ecologists ultimately want to compare and contrast such results for all of the roughly 600+ pairs containing sufficient data. We have applied these techniques to all of these pairs, and the resulting trend plots can be found online at `http://www.avianknowledge.net/content/toolbox/partial-dependency-plots/`.

CHAPTER 6

ON VARIABLE SELECTION, BIAS-VARIANCE, AND BAGGING *

The amount of noise which anyone can bear undisturbed stands in inverse proportion to his mental capacity.

— Arthur Schopenhauer

Chapter 5 looked at how to understand a large, complex decision tree ensemble that used almost all of the 200 predictor variables available. An alternative approach to improving model understandability is to start by learning a model with fewer inputs. Even if the resulting model is a complex function of the predictors, the model will be easier to study using black box analysis methods like sensitivity analysis (Breiman, 2001a) or partial dependence functions (Friedman, 2001) if it relies on fewer inputs. To this end, we ran forward stepwise variable selection to find the smallest set of predictors yielding excellent performance for a few selected bird species.

To our surprise, after 30 steps of variable selection model performances were still inferior to the performance when using all 200 predictors and the gap was closing slowly as more predictors were added (see Figure 6.1). Unlike most learning algorithms, bagging appeared to perform remarkably well with many noisy predictors.

This chapter examines how variable selection improves the accuracy of supervised learning through the lens of bias/variance analysis. We run variable selection for nineteen data sets and compare the bias-variance decompositions

*Reprinted with minor additions and revisions, with kind permission from Springer Science + Business Media:

Munson, M.A., and Caruana, R. (2009). On feature selection, bias-variance, and bagging. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2009, Proceedings, Part II* (eds. W.L. Buntine, M. Grobelnik, D. Mladenić & J. Shawe-Taylor), pp. 144–159. Springer-Verlag, Berlin Heidelberg, Germany.

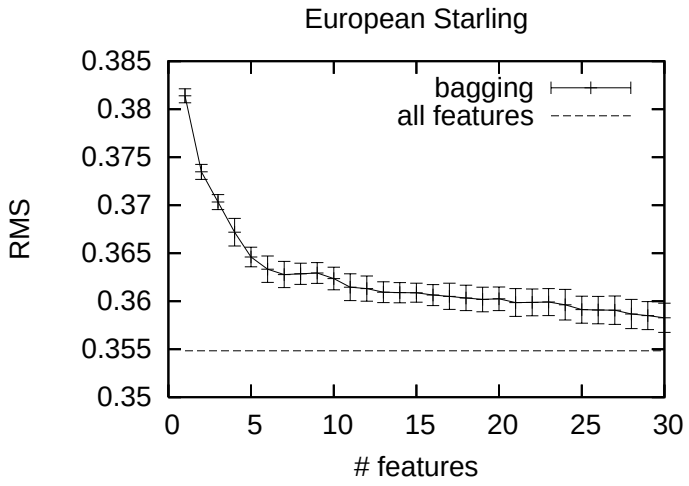

Figure 6.1: Bagging performance with forward stepwise feature selection. The *all features* line shows performance of bagging with all 200 features.

of single and bagged decision trees for many different variable subset sizes. The results show that the most accurate predictor sets correspond to the best bias-variance trade-off point, and this depends on the learning algorithm. Particularly with high variance algorithms such as decision trees, this is usually *not* the separating point between relevant and irrelevant predictors. With too many variables, the increase in variance outweighs the potential gains of adding (weakly) relevant predictors. When bagging is used, however, the increases in variance are small, which makes the reduction in bias beneficial for many more predictors. In many cases, the best bagging performance is obtained by using all available predictors.

While it is known that ensemble methods improve the base learner's ability to ignore irrelevant predictors (Ali & Pazzani, 1996; Bay, 1998), little is known about their effects on weak / noisy predictors. To explore this, we generate synthetic data and randomly damage varying percentages of the predictor values. The results show that bagging dramatically improves the ability of decision trees to profitably use noisy predictors.

6.1 Prior Work

6.1.1 Variable Selection and Bias-Variance Decomposition

Machine learning researchers informally recognized the relationship between the bias and variance of a learning algorithm and the number of predictors given to the algorithm years ago. For example, Kohavi and John wrote:

> From a purely theoretical standpoint, the question of which features to use is not of much interest. ... The optimal Bayes rule is monotonic, *i.e.*, adding features cannot decrease accuracy, and hence restricting a Bayes rule to a subset of features is never advised. In practical learning scenarios, however, we are faced with two problems ... The first problem is closely related to the bias-variance tradeoff [36,61]: one must tradeoff estimation of more parameters (bias reduction) with accurately estimating these parameters (variance reduction). (Kohavi & John, 1997, p. 276)

Their final summary remarks reinforce the connection between variable selection and algorithmic bias-variance tradeoffs:

> [Different] training set sizes might imply that a different set of features is optimal. If only a small training set is given, it may be better to reduce the number of features and thus reduce the [learning] algorithm's variance; when more instances are given, more features can be chosen to reduce the algorithm's bias. (Kohavi & John, 1997, p. 319)

The results below are the first empirical demonstration of this relationship. Surprisingly, the connection does not appear to be widely known: researchers fre-

quently describe variable selection as separating relevant from irrelevant predictors, and attribute performance improvements to the removal of irrelevant predictors (e.g., (Hand *et al.*, 2001, p. 362), (Reunanen, 2003, p. 1371), (Tuv *et al.*, 2006, p. 2181)). The language of relevant vs. irrelevant can be misleading because it implies a false dichotomy. As will be seen in Section 6.3, seemingly "irrelevant" predictors can become "relevant" by switching to a learning algorithm that is more robust to variance.

Some researchers have estimated bias and variance in the context of variable selection but typically only for the final variable set selected (e.g., Loughrey & Cunningham, 2005). Van der Putten and van Someran (2004) use bias-variance analysis to understand the wide performance spread of contestants in the 2000 CoIL challenge. They compare the bias-variance decompositions of a single subset (the top 7 predictors) against the original predictor set and find that variable selection is important for their problem (the decrease in variance outweighs the increase in bias).

6.1.2 Bagging and Variable Selection

Several pieces of work exist that address features (predictors) and bagging. We mention them here to avoid confusion and clarify the differences. (These techniques are not used in the experiments below.) First, *feature bagging* generates diverse simple models by training individual models with random samples of the features instead of (or in addition to) random samples of training examples (Ho, 1998; Bryll *et al.*, 2003), and is particularly useful for building ensembles with simple learners that are inherently stable (Bay, 1998). In *ensemble feature selection* (Opitz, 1999) multiple good feature sets are sought such that a) a good simple model can be built from each set, and b) the simple models are

maximally diverse from each other. Finally, *feature selection using ensembles* (Tuv *et al.*, 2006) uses statistics derived from tree ensembles to rank features. More generally, ensembles have been used in feature selection to find more stable feature subsets (Saeys *et al.*, 2008a; Tuv, 2006).

6.2 Methodology

6.2.1 Variable Selection Algorithms

We used two different variable selection algorithms to generate a ranking of predictors. On data sets with small to medium dimensionality (less than 200 predictors), we used forward stepwise variable selection (FSVS). On high-dimensionality data sets we used correlation-based variable filtering (CVF). (See Section 2.7 in Chapter 2 for descriptions of both algorithms.)

Each data set was divided into five folds. For FSVS, three folds were used for training, one for validation (to pick which variable to add), and one for testing final performance. For CVF, variable ranks were computed from the three training folds. The description for one data set, SATIMAGE, warns against using cross-validation; for that data set we used the given train/test split instead of cross-validation and pulled 435 examples from the train set to use as a validation set (about 10% of training).

6.2.2 Learning Algorithms

To handle the wide range of data sizes, we used two different decision tree packages. In all cases bagging used 25 trees per ensemble, and training samples were drawn with replacement.

For data sets with small to medium dimensionality (< 200 predictors), we used minimum message length (MML) decision trees implemented in Buntine's IND package (Buntine & Caruana, 1991). We selected MML trees because the Bayesian smoothing makes them relatively low variance, so in our experience the individual trees perform well and seem to be resilient to spurious and noisy predictors. Thus, they are less likely to require variable selection to achieve good performance (vs. a less sophisticated decision tree like ID3), making them a strong baseline method. At the same time, they are not aggressively pruned and are large trees, making them good candidates for bagging.[1]

For the high-dimensionality data sets, we used FEST[2], a decision tree package optimized for sparse data. To prevent overfitting, we tuned the maximum tree depth parameter in FEST to maximize performance of a single tree, using *all* predictors, on the validation fold of each data set. We tried depths of 1 through 10, and then the powers of 2 from 16 through 1024. The best performing depth was used for both single and bagged tree models.

A single FEST tree makes predictions from negative to positive infinity. We calibrated the predictions by fitting a sigmoid to convert them to probabilities (Platt, 2000). Validation data was used to fit the calibrating sigmoid.

6.2.3 Performance Metrics

Model performance was measured using zero-one loss and squared error. A loss of zero represents perfect prediction for these measures. Zero-one loss frequently has high variance, so MSE was used as the performance metric during FSVS when deciding which predictor to add.

[1] Experiments with other very different tree methods such as C4.5 (Quinlan, 1993) yield similar results.

[2] `http://www.cs.cornell.edu/~nk/fest/`

6.2.4 Data Sets

We used 19 classification tasks in our experiments: American Goldfinch presence / absence at bird feeders (AMEGFI), Lark Bunting presence / absence in the plains east of the Rocky Mountains (BUNTING), forest cover-type (COVTYPE), Pima Indians Diabetes (PIMA), letter recognition (LETTERS), mushroom identification (MUSHROOM), land classification from satellite images (SATIMAGE, Statlog dataset), sonar classification (SONAR), soybean disease classification (SOYBEAN), spam detection (SPAMBASE and SPAMTREC[3]), cardiac abnormalities (SPECTF), hyper-thyroid conditions (THYROID), two medical prediction tasks (MEDIS and MG5), protein crystallography diffraction pattern analysis (CRYST[4]), hand-written digit recognition (DIGITS[5]), real vs. simulated auto racing and aviation text categorization (REAL-SIM[6]), and finding translation initiation sites (TIS[7]).

Table 6.1 summarizes the data sets; high-dimensional data sets are listed below the line along with the maximum tree depth(s) chosen during parameter tuning. Data sets were chosen to cover a range of sizes (number of examples) and dimensionalities (number of predictors). We used the first 30,000 points from the COVTYPE data set to make the experiments more affordable.

[3]Created from TREC 2005 Spam Public Corpora. Nikos Karampatziakis, personal communication.

[4]`http://ajbcentral.com/CrySis/dataset.html`, unscaled version

[5]MNIST data set converted to binary classification (class 0 = digits 5 or below; class 1 = rest). Original available from `http://yann.lecun.com/exdb/mnist/`

[6]`http://csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`

[7]`http://datam.i2r.a-star.edu.sg/datasets/krbd/` (Kent Ridge Biomedical Data Repository)

Table 6.1: Summary of datasets.

Data Set	# Samples	# Predictors	# Classes	Max Depth
amegfi	23948	195	2	
bunting	20998	175	2	
covtype †‡	30,000	54	7	
letters †	20,000	16	26	
medis	14199	63	2	
mg5	22157	100	2	
mushroom †	8124	22	2	
pima †	768	8	2	
satimage †	6535	36	6	
sonar †	208	60	2	
soybean †	683	35	19	
spambase †	4601	57	2	
spectf †	266	44	2	
thyroid †	3772	27	5	
cryst	5,498	1341	2	4
digits	70,000	779	2	16–1024
real-sim	72,309	20,958	2	256–1024
spamtrec	87,688	405,915	2	256–1024
tis	13,375	927	2	5–6

†: Available from UCI Machine Learning Repository (Asuncion & Newman, 2007).
‡: First 30,000 examples from full data set.

6.2.5 Bias-Variance Decomposition

Bias and variance were empirically estimated using the method detailed in Section 2.4.2. To improve the estimates of bias and variance, we repeated the variable selection and bias-variance estimates for each of five folds and averaged the results.

In the results below we focus on the decomposition for squared error because variable selection hill climbing used MSE. We did, however, compute the bias and variance of zero-one loss; the results were qualitatively identical to those obtained using the squared error decomposition.

6.3 Bias-Variance of Variable Selection

We estimated the bias-variance decomposition for all the data sets in Table 6.1 at multiple predictor set sizes, for both single and bagged decision trees. Variable subset orderings were found using forward stepwise variable selection (FSVS, top of table) and correlation coefficients (CVF, bottom of table). FSVS evaluated performance using single trees or bagged trees, to match the algorithm used in the final comparison. After establishing subset orderings (and tuning the maximum tree depth for the high dimensional data sets), the training and validation sets were pooled as described in Section 2.4.2.[8] Bias-variance estimates were made at several points along the subset ordering sequence. The entire experiment was repeated across 5-folds and the 5 estimates averaged.[9]

The results cluster into two categories. Figure 6.2 shows representative results for two of the data sets. The *total* height of each bar is the error for the number of predictors on the x-axis. The pair of bars for each number of predictors correspond to using a single tree (left in pair) and using bagged trees (right in pair). Each bar is subdivided into portions that are due to a) the variance of the algorithm, and b) the bias of the algorithm. The bias portion also contains any noise inherent in the domain. For comparison's sake, results are shown for both mean squared error (left column) and zero-one loss (right column). For the moment we focus on patterns in the total error. Detailed observations about bias and variance are below.

Variable selection does not improve the performance of single or bagged trees on data sets in category one. Consider the graphs for COVTYPE (top row of

[8]When validation data was needed to calibrate predictions, we set aside 10% of the training sample drawn from the pooled data. Thus, the calibration data varied with each training sample.

[9]The PIMA, SONAR, SPECTF, and THYROID data sets exhibited substantial variance in the results, so we repeated the 5-fold cross-validation five times using different seeds to divide the data into folds.

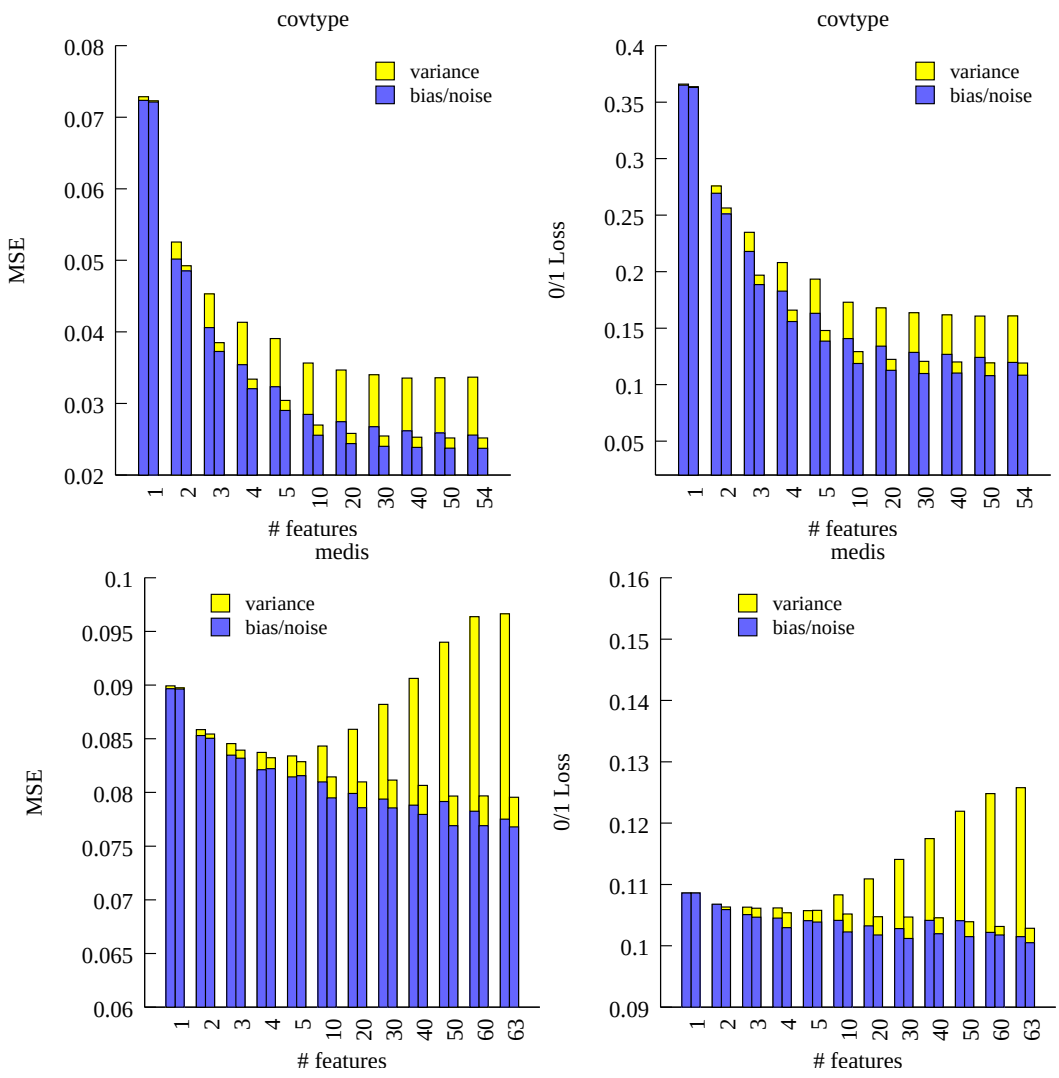

Figure 6.2: Bias-variance decomposition of squared error and zero-one error for typical data sets. Left bar in pair: *single tree*; right bar: *bagging*. To better show interesting parts of graphs, the y-axes do not start at 0.

Figure 6.2). Both bagging and the single tree perform as well (or better) using all predictors (right side of graph) than when using a subset (interior of graph). The graphs in Figure 6.3 show qualitatively similar results: variable selection does not improve the accuracy of single or bagged trees. (The results for zero-one loss are qualitatively the same as for squared error, and are omitted for most of the data sets.)

The second category, however, contains data sets on which variable selec-

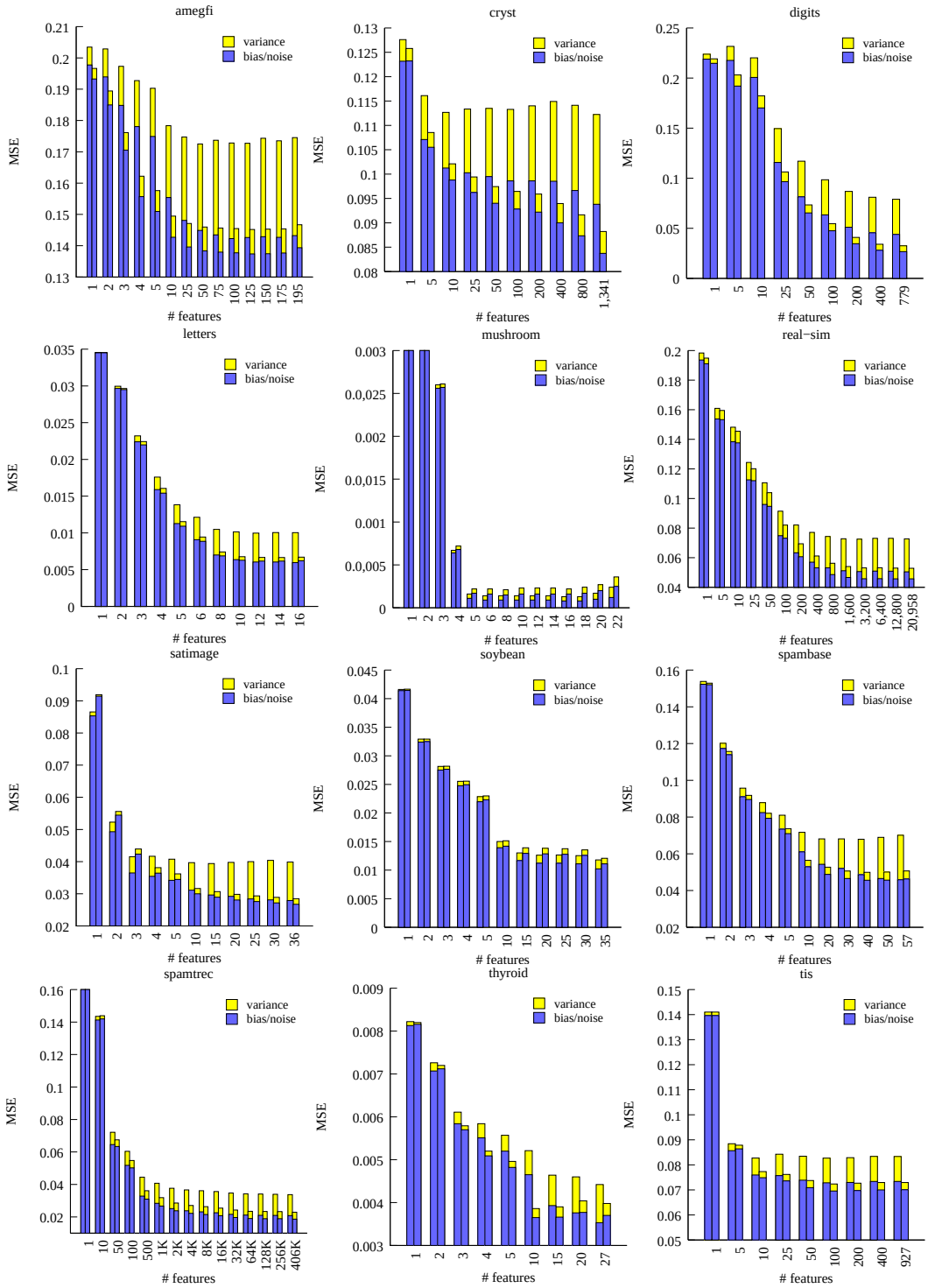

Figure 6.3: Bias-variance decomposition of squared error for variable selection on data sets where variable selection does not improve performance (category 1). Left bar in pair: *single tree*; right bar: *bagging*.

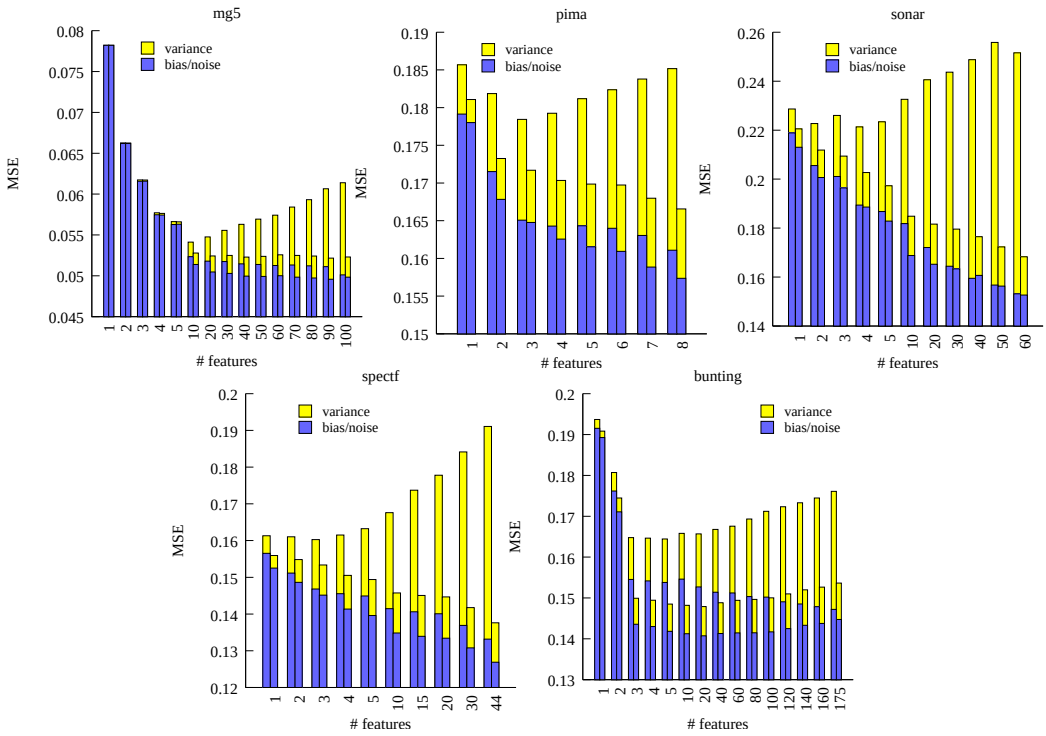

Figure 6.4: Bias-variance decomposition of squared error for variable selection on data sets where variable selection helps single trees (category 2). Left bar in pair: *single tree*; right bar: *bagging*.

tion improves single tree accuracy but does not improve bagging's accuracy. Looking at MEDIS (bottom row of Figure 6.2), the single tree achieves the minimum loss between five and ten predictors. Bagging, on the other hand, first reaches its minimum loss around 50 predictors, at which point the loss flattens out and stays roughly constant. The graphs in Figure 6.4 (*except* BUNTING — see discussion below) contain similar results. While single trees eventually lose performance as more predictors are added, bagging maintains or improves its performance with more predictors.

It is worth noting that for data sets in both categories (CRYST, LETTERS, MEDIS, PIMA, SATIMAGE, SONAR, SPAMBASE, SPECTF, TIS), bagging performance continues to improve as more predictors are added after the performance of single trees has plateaued (category 1) or peaked (category 2). In other words,

bagging performance flattens further to the right in the graphs. Bagging seems to be capable of extracting information from noisy predictors as well as ignoring irrelevant ones. Section 6.4 explores this issue further.

For all the data sets, bias decreases as more predictors are added. This makes intuitive sense since extra predictors can be thought of as extra degrees of freedom. The decrease is largest for the first few predictors; after that, the bias levels off as the algorithms become sufficiently flexible. Although the bias error is very similar for single trees and bagged trees, bagging does sometimes reduce bias slightly. This corroborates findings in other studies (Bauer & Kohavi, 1999).

Counter to bias, variance increases with the number of predictors. However, this effect is much stronger for single trees than for bagged trees. Whereas the variance for bagging quickly asymptotes to a small amount, the variance for single trees grows quickly and may not asymptote. This is bagging's primary advantage.

In data sets where the performance of single trees levels off (e.g. COVTYPE), the algorithm's bias and variance asymptote so that adding more predictors does not hurt. Usually bagging's variance stabilizes earlier and to a lower amount, which allows it to reach lower error and benefit from additional bias decreases as more predictors are added.

In data sets where the single tree performance gets worse with too many predictors (e.g. MEDIS), the variance increases outstrip the initial benefits of reduced bias. This rarely happens to bagging because its variance typically asymptotes to a small amount of error.

Figures 6.3 and 6.4 contain three anomalies, one large and two small. The most important anomaly is the graph for BUNTING, which does not fit into either category described above. On this data set, both single trees and bagging

hit peak performance between 5 and 20 predictors, after which their performance degrades. Thinking that perhaps this domain was just extremely noisy and that more averaging would eliminate bagging's overfitting, we re-ran variable selection on one fold using 100 trees instead of 25. With 100 trees, bagging's performance was slightly better at all points along the x-axis (compared to bagging with 25 trees), but still overfit past 20 predictors and to the same degree (Fig. 6.5). Further investigation revealed that this data set contains several predictors that can be combined to create semi-unique identifiers for individual examples in the training set. All the trees in the ensemble effectively memorize these identifiers and then do poorly on the validation and test data. This can be seen in the bias-variance decomposition. Although the variance has asymptoted, the bias for bagging stops decreasing and begins increasing. With the extra predictors, the trees in the ensemble more consistently construct unique identifiers for training examples and lose diversity in their incorrect predictions.[10]

The other two anomalies are that single decision trees perform (slightly) better than the bagged tree ensemble for the MUSHROOM and SOYBEAN data sets. For MUSHROOM, the single tree is extremely confident in the class probabilities it assigns, and always picks the right class (zero-one loss is 0%). Bagging also always picks the right class, but the randomization from sub-sampling the training data plus averaging results in *slightly* less confident class probabilities (probability mass is pushed away from the extremes). This small bias away

[10]A more detailed explanation follows. The task in BUNTING is to predict the presence or absence of a Lark Bunting. Data are collected at multiple sites; in particular, repeat observations are made at sites over time. Identifying the site is incredibly useful for predicting presence or absence, but is not ecologically interesting. Thus, the five data folds were partitioned by site (i.e. all examples from a site appear in a single fold). Most predictors are tied to location (e.g. habitat), so the decision trees can easily learn to map inputs to sites in the training set using only a few predictors. Trees that do this make bad predictions on the validation and test folds. If the folds are created by assigning examples to folds instead of sites (spreading sites across train/valid/test), bagging does not overfit while a single tree does.

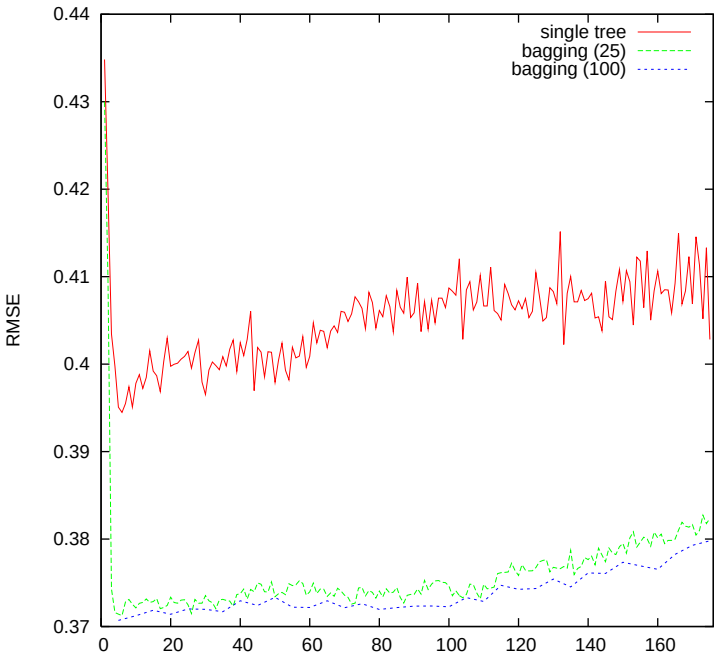

Figure 6.5: Effect of adding more bags for the BUNTING data set. The graph shows root mean squared error (y-axis) of single and bagged decision trees as a function of number of predictors (x-axis). Even with 100 bags, bagged trees overfit after a few predictors. It is unlikely that the extra error from adding predictors is due purely to variance that could be reduced by more averaging.

from extreme values has a small effect on squared error. SOYBEAN has a different problem. This small data set has 19 classes. Cross-validation and bagging sampling reduces the number of cases for some classes in the training samples so that probability estimates become less reliable and MML pre-pruning prevents leaf expansions, yielding trees that are too small.

Throughout this section (and most of the chapter), noise and bias have been conflated since we do not have a way to separate them on real data. We hypothesize that the large decreases in bias—coinciding with adding the first few predictors—is partly due to decreases in noise. Intuitively, the Bayes optimal error rate, given only a single predictor as an information source, may be quite bad (effectively high noise). As more information becomes available (more predictors), the Bayes rate should improve as uncertainty decreases.

To summarize this section, these graphs show that bagging is resilient to noisy predictors. *Variable selection usually is unnecessary to get good performance from bagged models.* Further, picking the best subset size (using cross-validation, for example) *is not* equivalent to choosing the informative predictors and discarding irrelevant predictors. Rather, a discarded predictor may be weakly informative (or correlated with a predictor already selected) but cause too much variance when selected for the extra information to improve accuracy. The fact that discarded predictors are sometimes informative was previously noted and exploited to improve model accuracy by using discarded predictors as extra model outputs during training (Caruana & de Sa, 2003).

6.4 Noisy Informative Predictors

In the experiments above, bagging's performance continues to improve after the single tree's performance peaks or plateaus. This suggests that ensemble methods are not only resilient to irrelevant predictors (Ali & Pazzani, 1996), but also better able to take advantage of predictors containing useful but noisy information.

We generated synthetic data to study whether bagging improves the base learner's ability to use weak predictors. A binary classification problem was derived from the equation:

$$v = X_1 + X_2 X_3 + X_4^2 + \text{sign}(X_5 + X_6)$$

The class label is 1 when $v \geq 0$, and 0 otherwise. Each X_i is a univariate Gaussian variable with 0 mean and unit variance. The $\text{sign}(z)$ function returns 1 if $z > 0$ and -1 otherwise. This function was chosen to be challenging for decision tree learning algorithms.

We generated 5,000 examples using the above function, randomly corrupted some of the inputs to generate weak predictors, split the data set into 5 folds, and ran a bias-variance analysis using the procedure outlined in Section 2.4.2. A predictor was corrupted by permuting a fraction of its values, chosen randomly among the examples. For example, at the 0.1 corruption level, 10% of the values in corrupted predictors are shuffled. This was repeated 20 times, creating 20 noisy versions of each corrupted predictor. Half of the X_i predictors were corrupted, independently of each other, while the other X_i were left intact. Single decision trees and bagged decision trees were trained using the intact predictors and the noisy duplicates, but not the original versions of the corrupted predictors. To avoid experimental bias, this process was repeated for all $\binom{6}{3}$ combinations of choosing 3 predictors to corrupt, and the results averaged.

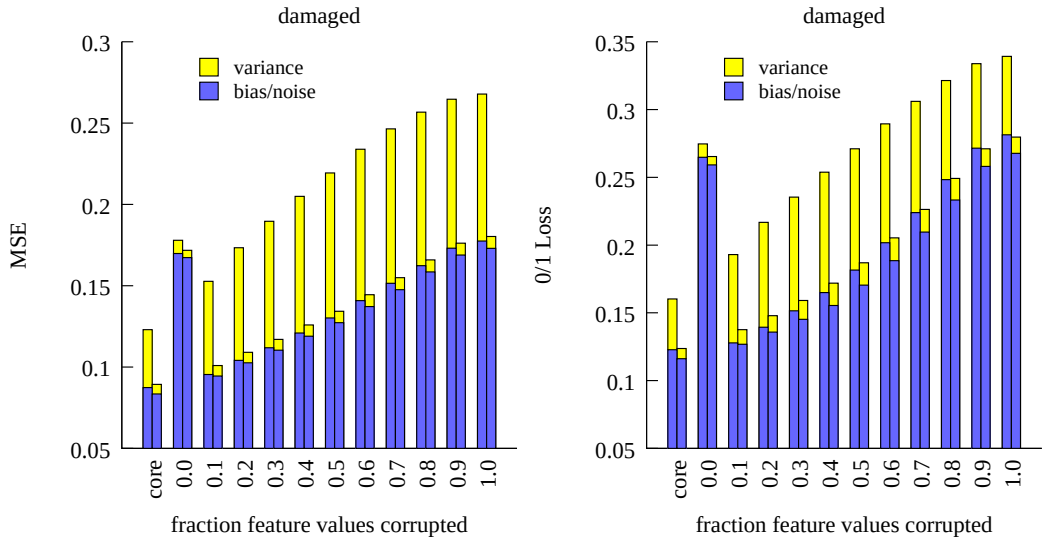

Figure 6.6: Bias-variance decompositions for DAMAGED data sets with corrupted predictor values. Left bar in pair: *single tree*; right bar: *bagging*. Note that the y-axes do not start at 0.

Figure 6.6 shows the results for different corruption levels. The far left column, *core*, is the error obtained when training using only the unblemished 6 original predictors, and shows the best performance obtainable on this data set

for these algorithms (i.e. when the ideal predictor set is used). The 0.0 column shows the performance obtained using only the 3 intact predictors, without any corrupted predictors. Performances that beat this baseline indicate an algorithm is learning something useful from noisy predictors. Finally, the far right column (1.0) shows the performance when the corrupted predictors are pure noise (irrelevant predictors).

We make the following observations. First, at low corruption levels both single and bagged trees learn something useful from the noisy predictors. For bagged trees, performance is close to that of using the ideal predictor set. Second, noisy predictors increase the bias (because noise is lumped in with bias in our empirical decomposition) of both single and bagged trees (vs. core), and increase the variance of single trees. Third, the main effect of increasing the corruption level is to increase the bias/noise component. Finally, the extra variance in the single trees means that the benefits of noisy predictors are quickly lost as the corruption level increases. At least for this synthetic task, the problem is more pronounced for squared error. In contrast, the bagged trees are remarkably resilient to damaging the predictor values, and are able to extract useful information when as much as 80% of the values are corrupted.

6.5 Conclusions

Our experiments show that variable selection finds the predictor set that represents the best trade-off between the bias of having too few predictors and the variance of having too many predictors. Because of this, most variable selection algorithms are not reliable methods for determining which predictors are relevant and irrelevant to a given problem: the threshold for predictor inclusion/exclusion depends on the learning algorithm. Ultimately this limits

the utility of variable selection for discovering which factors are important and unimportant in problems such as the avian analysis that originally motivated this work.

A by-product of our analysis is the discovery that when variable selection is too expensive to be feasible or effective, bagging provides a viable alternative to protect from the overfitting that can occur when models are trained with too many predictors.[11] The bagged models always benefit from using at least as many predictors as the individual unbagged models. In fact, when models will be bagged, any amount of variable selection often is detrimental, and it is better to train the base models using all available predictors. One interpretation of our results is that variable selection is best viewed as a model regularization method instead of as a means of distinguishing relevant from irrelevant inputs.

[11]Of course, variable rankers like CVF are less expensive than training a single bagged tree ensemble. Even with a ranker, however, there is the problem of choosing a cutoff threshold for which predictors to include — which typically requires training models for multiple candidate threshold levels. Thus, carefully choosing a threshold could easily cause a variable ranker to be more computationally expensive than training a single ensemble.

CHAPTER 7

CONCLUSIONS

Ne'er look for birds of this year in the nests of the last.

— Miguel de Cervantes, *Don Quixote de la Mancha*

Analysts, scientists, and companies have been slow to adopt machine learning and data mining techniques to make sense of the deluge of data being collected in seemingly all disciplines—despite the existence of many effective algorithms for learning models from data. The problem is that many steps are necessary to get ready to run a machine learning algorithm, and more steps are needed *after* learning a model to figure out what to do with the model. Most practitioners who use machine learning and data mining techniques believe that all of these steps are important to finding successful solutions to real problems, yet far less research energy is spent on these pre- and post-processing steps (see section 1.4 for details).

As a step towards facilitating the use of machine learning, this dissertation has studied three specific challenges that occur before applying machine learning and one that occurs after applying machine learning.[1] After summarizing the dissertation's findings (§ 7.1), we list open problems related to the modeling pipeline (§ 7.2).

7.1 Summary of Findings

Chapter 3 proposed cross-data validation, a framework for quantifying the quality of a data set relative to a second, benchmark data set whose quality is

[1] The research on understanding decision tree ensembles in chapter 5 deals with understanding a model that is already built, but can also have an impact on future iterations of models by informing decisions about how to change data representations.

known. The method was developed to solve a problem from ecological modeling: how trustworthy are the high volume data sources that collect bird observations from volunteers using low-structured collection protocols (i.e., protocols with few rules or restrictions for how data can be collected)? We showed that data sources can be compared by first training models from each data source and then comparing the predictions from the models on test data. The models generalize beyond the locations and times when data were collected and enable comparisons between data sources with non-overlapping data examples. Varying the amount of training data simulates the effect of collecting more data and allows estimates of the efficiency of the different data sources. In addition to verifying the quality of new data sources, this information is also valuable for assessing the suitability of competing data sources for answering a particular modeling question. We applied the methods to show that data from eBird, a low-structured data source, contained information comparable to that from the North American Breeding Bird Survey, a highly-structured data source monitoring bird populations in the summer. Since eBird collects year-round observations, establishing its validity enables future ecological studies of bird populations during migration and wintering seasons—seasons for which large-scale data has not been available.

In chapter 4 we evaluated an ensemble-based approach for learning models optimized for arbitrary performance measures in the domain of natural language processing. The approach, called ensemble selection, incrementally selects pre-trained base models from a model library to greedily optimize a user-supplied performance measure. Our empirical results showed that the ensemble selection *framework* is useful for automating the important modeling steps of algorithm and parameter selection, removing the need for human expertise in

choosing algorithms and parameter settings. Exploring many algorithms and carefully tuning parameters was surprisingly effective for optimizing a given performance measure. Compared to simple algorithm and parameter tuning, ensemble selection improved performance for two of the three NLP tasks studied for all but one performance measure (GACC). In contrast, ensemble selection overfit on the third task (PSF Hierarchy) and produced *worse* performance than the best model for several performance metrics. More work is needed to understand when ensemble selection can be safely used.

While ensemble models are usually among the best performing models for a given task, they are nearly impossible to understand. Chapter 5 presented fast statistics for finding the most important predictors used in a decision tree ensemble. We applied the tree statistics to models of species occurrence and showed that the statistics correlated well with sensitivity analysis, particularly for ranking the top 25 most important predictors (out of 197 total predictors). While sensitivity analysis is more general (it can be applied to any kind of model), the tree statistics scale better with large data sets. For the species occurrence models studied, the statistics were 500 times faster than sensitivity analysis. We gave examples of how to visualize the effect of the important predictors on the occurrence model using partial dependence plots (Friedman, 2001). Correlated predictors can cause masking and dilution effects that result in not finding all important predictors. We explored expanding the set of important predictors by computing the association between the top predictors and all other predictors. Alternate predictors sets were considered by swapping in highly associated predictors and training new models from the resulting predictor sets, allowing ecologists some freedom to choose the most biologically plausible predictor variables while still maintaining accurate models. The swapping results

showed that even the top 20 most important predictors contained redundant information, and some could be removed without hurting performance. Future work exploring alternate predictors should be sure to start by identifying a minimal set of predictors.

Chapter 6 studied the interaction of variable selection and bagging through the lens of bias-variance decomposition. We showed that variable selection finds a predictor subset with a good bias-variance tradeoff. In other words, variable selection does not separate the relevant and irrelevant variables (as commonly explained in machine learning literature); rather variable selection finds the predictors that the learning algorithm is able to use and discards the rest. Consequently, the best tradeoff point will be different for different algorithms. Because variable selection improves performance by reducing variance and overfitting, it can be viewed as a regularizer. As such, it is usually redundant for learning methods like bagging that have low variance and do not require regularization. We showed that bagging copes well with lots of features not only because it is resistant to large numbers of irrelevant predictors (Ali & Pazzani, 1996) but also because it improves the base learner's ability to learn from weak diluted predictors. If good performance is the only criterion for model success, there is no need to apply variable selection to pre-process data before training a bagged model.

7.2 Open Problems

There are a number of interesting open problems raised by our results.

1. What are the prerequisites for ensemble selection to build an ensemble that performs at least as well as the best single model (after parameter tuning

and algorithm selection)? Ensemble selection has produced extremely accurate models on many problems (Caruana *et al.*, 2004, 2006b; Niculescu-Mizil *et al.*, 2009), but sporadically hurt performance in our experiments in chapter 4. It would be nice to have rules of thumb to help practitioners decide whether ensemble selection is likely to work for a given problem. Similarly, are there theoretical bounds for how much it can hurt? Intuitively it seems that the constructed ensemble should never be worse than the worst model in the model library. This limit is both unproven and loose. One would hope for stronger guarantees.

2. Are there fast importance measures for non-tree models like non-linear support vector machines (particularly for RBF kernels), neural networks, and graphical models (e.g., conditional random fields, markov random fields)? Like decision trees, these model classes often have structure that might be exploited.

3. Can we design importance measures that correct for masking and/or dilution effects?

4. Can the information structure in a set of correlated predictors be extracted to find all of the interchangeable predictor sets? Ideally this would produce something like a menu for the information relevant to the task. For example, accurate predictions might require inputs that cover three distinct information needs, and there might be multiple ways to satisfy each need. The first information need might be filled by including predictor A or by including both predictors B and C.

5. Do the results of our study on variable selection, bias-variance, and bagging hold for other ensemble learning algorithms? We suspect that the results will hold for random forests, an algorithm that is similar to bagging.

Boosting, on the other hand, is known to sometimes overfit, and removing noisy predictors through variable selection may improve performance on some tasks by reducing opportunities to overfit the training data. (Fewer predictors means fewer degrees of freedom for learned models.) It would be particularly interesting to see how boosting responds to the corrupted predictor experiment from section 6.4.

6. What are the limits to bagging when data includes large numbers of predictors? Daria Sorokina (personal communication) has reported seeing bagging significantly overfit on problems with noisy data and large numbers of predictors. She found that training using only the most important predictors improved performance significantly. Part of the problem seems to be specific to decision trees and how they run out of training data during recursive partitioning.

Within the domain of species distribution modeling, and ecological modeling in general, are a number of open problems that are both computational and ecological.

1. How can we build an infrastructure that supports large-scale modeling given the large size of the data sets involved and that predictors are derived from multiple geographically separated databases? Instead of aggregating and collecting all the predictors in a single data warehouse, should we instead move the computation to the various data stores? What changes to learning algorithms are needed to support federated computation in which learning is distributed across different sites?
2. How can models efficiently incorporate time-varying predictors like weather and current canopy cover?

3. Is there an ecological theory to guide scientists and modelers in choosing appropriate spatial scales for studying/modeling a species?
4. How can models be learned that are "scale-free" and can make predictions at multiple spatial and/or temporal scales?
5. The ecological community generally agrees that modeling species abundance is harder than modeling species occurrence (Stauffer, 2002, p. 59). Are there modeling improvements that can generate accurate abundance models? Alternatively, maybe abundance is inherently hard, and we should not expect to learn extremely accurate models. But are current models even close to optimal? At the moment, there is no ecological theory for the inherent uncertainty of ecological processes and systems, and as a result we do not have good upper bounds for which to strive.

In addition to these specific problems, a great deal of research is still needed to automate the various modeling stages and make them less challenging to potential users of machine learning and data mining methods. A useful exercise is to consider implementing a toolkit for machine learning that would be usable by analysts and scientists with minimal training (e.g., perhaps a week of training). What obstacles block the development of such a useful toolkit?

The following is a partial list of "big picture" problems that hinder the easy use of machine learning and data mining tools.

- Almost as a rule, there are no confidence intervals for the predictions an automatically learned model makes or for the model itself. Reliable confidence bounds are crucial if a policy maker is going to take action based on the results of machine learning. Empirical confidence bounds on predictions can be estimated by training multiple models from different training

data subsets and studying the distribution of predictions from the different models. This is computationally expensive, however, unless the model is fast to train.

- There is a wealth of prior domain knowledge in most scientific disciplines—so much knowledge in fact that an expert in any given field increasingly knows only a fraction of the facts and theories from his field. And yet, the typical machine learning algorithm starts *tabula rasa*, with a blank slate, when building a model to explain or predict the training data. An important exception are graphical models, in which an expert can hand build the model structure to reflect prior domain knowledge—but this process is time consuming, requires an expert, and assumes that the expert knows all of the relevant prior knowledge. Is there a way to initialize the learning algorithm by automatically extracting domain knowledge from published research papers?

- Chapter 5 talked about the opaqueness of learned models, but the process of building that model is itself opaque and hard to understand. Why did the learning algorithm build *this* particular model? The answer to this question would be extremely helpful for troubleshooting models. If learned models could justify their predictions, users would both be able to fix problems (e.g., by adding new predictors or constraining the algorithm with domain knowledge) and sanity check the predictions. The latter would be useful for deciding what actions should be taken in response to predictions.

- There is no theory to guide users (or a semi-automated toolkit) to pick an appropriate learning algorithm for a given task. Different learning algorithms have different strengths and advantages. Currently, machine learn-

ing experts choose an algorithm based on prior experience, rules of thumb, and intuition about what seems to fit the task at hand. To facilitate wide spread use, we need theories that predict which algorithm(s) are likely to work well based on easily computable data characteristics.

- Different algorithms require different data transformations to work well. Like the skill of picking a good algorithm for a given problem, there is no theory to help users prepare their data for a particular learning algorithm. Currently, practitioners rely on folklore (e.g., rescale all predictors to have mean zero and unit variance when training a neural network) and prior experience. A theory for matching data transformations to algorithms would enable automated transformations and simplify applying multiple learning algorithms to a data set.
- How should data be transformed to facilitate analyzing learned models? There is a long history in machine learning of manipulating data to facilitate modeling, but the question of preparing data to facilitate analyzing computational results is less well understood. Let a **factor** be an input that is created by transforming and combining one or more of the measured predictors. An analyst studying a model might desire many criteria in addition to the need to have an informative set of factors: a small number of factors, independent factors, factors that are conceptually tight and align with previous domain knowledge (e.g., a factor computed from weather predictors is easier to reason about than a factor computed from weather and habitat and population density predictors), and "simple" factors that are computed from a small (sparse) set of predictors. There are tensions between these desiderata, and the tradeoffs are not well understood.
- An easy to use toolkit should issue warnings about data points that are

outliers. Integrating data cleaning with model learning would help less experienced users avoid drawing conclusions from models built on bad data.

- Learning algorithms should be more *self-aware* about what they do and do not know. What kinds of examples is the learning algorithm (and hence, the model) uncertain about? What examples is the algorithm most curious about? What data would the algorithm choose to collect if it could? While some algorithms may have some self-awareness, users typically do not have access to this information. How can the learning algorithm and user interact to make data collection more efficient?
- Instead of discarding predictors that contain redundant information, robust learning algorithms should take advantage of the multiple information sources to produce models that can perform well even if some inputs are unavailable or are corrupted. By way of analogy, when a person drives a car they rely on many clues to stay on the road and avoid crashing: the center line, the line separating the lane from the shoulder, looking into the distance, etc. If one of the lane lines is obscured, a human driver can still stay in his lane. In contrast, most automatically learned models will pick the most useful predictor (e.g., the line between the lane and the shoulder) and discard the others; if that predictor gets corrupted or is unavailable, the model will fail in unpredictable ways. While some work has been done on robust learning (O'Sullivan *et al.*, 2000), more work is needed.

In closing, I believe that the machine learning community has a good understanding of how to learn models from data, as evidenced by the proliferation of successful learning algorithms and by the successful application of standard algorithms to new domains (like ecological modeling). The challenge now facing

the community is making these algorithms and the models they can learn usable by—and therefore useful to—analysts and scientists without backgrounds in machine learning.

APPENDIX A

SURVEY OF IMPORTANCE AND DIFFICULTY OF MODELING STEPS

This appendix describes the survey used to collect data about the importance and difficulty of different modeling steps. Chapter 1 reports the main survey results. Auxilliary results are reported below next to the corresponding survey question.

A.1 Survey Distribution and Collection

The survey was conducted online. Respondants were solicited through two email lists: ml-news@googlegroups.com (a news forum for the machine learning community), and KDnuggets (an online news letter for the data mining community). Responses were collected for 1.5 weeks after the survey announcement was posted. Figure A.1 contains the text of the survey announcement.

Twenty-four respondants completed the survey. Two respondants used dummy values for all questions (e.g., zero percent time spent during all modeling stages); one of these commented that he/she simply wanted to view all survey questions. (Survey questions were split across multiple web pages.) I discarded these two responses, leaving 22 survey responses.

A.2 Survey Questions and Answers

The survey was split into four pages. Survey takers could not return to pages they had already completed. Answers for each question are italicized below.

Page 1: Your Background

1. How many *completed* systems have you worked on where data mining or machine learning were important to success?

There are many steps required to build and deploy a system with a significant machine learning or data mining component. I am interested in how much time is spent per step in developing real systems and the community's opinion of the importance of the various steps.

As part of my dissertation I am conducting a short survey (5–10 minutes) of the community's experience with developing real systems. The survey can be found at:

`http://www.cs.cornell.edu/~mmunson/model-step-survey.html`

I would be grateful to anyone who can spare a few minutes to take the survey. Only aggregated statistics will be published. I will tabulate and post survey results at the above web page on March 6.

Thank you for your time, and I look forward to reading your survey responses.

Sincerely,
Art Munson
Cornell University

Figure A.1: Survey announcement email.

A completed system is either deployed or results in significant contributions to a domain outside of computer science (e.g., a publication in non-CS journal).

Number of systems (frequency): 1 (6), 2 (4), 3 (5), 4 (2), 5 (1), 7 (1), 8 (1), 10 (1), 40 (1).

2. (Optional) Please list key words or phrases that describe your interests, expertise, and/or background. One phrase or key word per line.

   Suprisingly, almost all respondants completed this question. The answers were highly varied and included domains like medicine, robot control, natural language processing, customer modeling and retention, advertising, and finance.

Page 2: Difficulty and Importance of Modeling Steps This page asks questions about your experience with the following modeling steps:

- Data Collection (not raw data collection, but any work team did to gather data into hands of analysts)
- Data Preparation (e.g., data integration and fusion, data cleaning, handling missing values)

- Change Data Representation (e.g., rescaling and normalizing features, transforming prediction target, feature selection, dimensionality reduction)
- Learning a Model from Data (e.g., posing the problem, algorithm selection or design, hyper-parameter selection)
- Performance Evaluation (accuracy and confidence in predictions)
- Study Model (e.g., to understand the model, to discover knowledge about domain theory, or to identify regions where model makes risky extrapolations)

1. Choose one system you have worked on with a modeling component (most recent, biggest, most successful, etc.). Estimate the percentage of time spent in each modeling step. Please include both your effort and your collaborators' efforts, but omit computer time. Rough estimates are sufficient.

   Results can be found Figure 1.2.

2. How important was each step to the success of the system in the previous question? [Respondant chose one of following for each step: not important, slightly important, moderately important, important, or critically important.]

   Results can be found in Figure 1.3.

Page 3: Focus of Research Community

1. In your opinion, how much energy does the *research community* spend addressing each modeling step? [Respondant chose one of following for each step: negligible energy, a little energy, moderate energy, lots of energy, or enormous energy.]

   Results can be found in Figure 1.4.

Page 4: Thank you! Thank your for completing the survey. If you have any extra comments or feedback you may leave them in the box below.

1. Additional Comments / Feedback (optional):

   Only 10 respondants left comments. Two said they found the survey interesting. Six respondants left comments elaborating why they felt certain modeling step(s) were the most important for success in the domains they worked in.

BIBLIOGRAPHY

Abazajian, K.N. *et al.* (2009) The seventh data release of the sloan digital sky survey. *The Astrophysical Journal Supplement*, **182**, 543–558.

Abe, N., Zadrozny, B. & Langford, J. (2004) An iterative method for multi-class cost-sensitive learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. W. Kim, R. Kohavi, J. Gehrke & W. DuMouchel), pp. 3–11. ACM, New York, NY, USA.

Ali, K.M. & Pazzani, M.J. (1996) Error reduction through learning multiple descriptions. *Machine Learning*, **24**, 173–202.

Asuncion, A. & Newman, D. (2007) UCI machine learning repository. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Bagga, A. & Baldwin, B. (1998) Algorithms for scoring coreference chains. In *Linguistic Coreference Workshop at 1st International Conference on Language Resources and Evaluation* (eds. S.J. Shelton, E. Hovy & V. Raskin), pp. 563–566. European Language Resources Association (ELRA), Paris, France.

Baillie, S.R., Balmer, D.E., Downie, I.S. & Wright, K.H.M. (2006) Migration Watch: An Internet survey to monitor spring migration in Britain and Ireland. *Journal of Ornithology*, **147**, 254–259.

Bart, J., Hofschen, M. & Peterjohn, B.G. (1995) Reliability of the breeding bird survey: Effects of restricting surveys to roads. *The Auk*, **112**, 758–761.

Bart, J. & Schoultz, J.D. (1984) Reliability of singing bird surveys: Changes in observer efficiency with avian density. *The Auk*, **101**, 307–318.

Bartlett, J. & Kaplan, J. (eds.) (1992) *Familiar Quotations*. Little, Brown and Company, 16 edn.

Bauer, E. & Kohavi, R. (1999) An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, **36**, 105–139.

Bay, S.D. (1998) Combining nearest neighbor classifiers through multiple feature subsets. In *Proceedings of the 15th International Conference on Machine Learning* (ed. J.W. Shavlik), pp. 37–45. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Becker, P.H. (2003) Biomonitoring with birds. In *Bioindicators & Biomonitors: Principles, Concepts and Applications* (eds. B.A. Markert, A.M. Breure & H.G. Zechmeister), pp. 677–736. Elsevier. Trace Metals and other Contaminants in the Environment 6.

Billman, D. (1996) Structural biases in concept learning: Influences from multiple functions. *The Psychology of Learning and Motivation*, **35**, 283–321.

Billman, D. & Davila, D. (1995) Consistency is the hobgoblin of human minds: People care but concept learning models do not. In *Proceedings of the 17th Conference of the Cognitive Science Society* (eds. J.D. Moore & J.F. Lehman), pp. 188–193. Lawrence Erlbaum Associates, Mahwah, NJ, USA.

Blum, A.L. & Langley, P. (1997) Selection of relevant features and examples in machine learning. *Artificial Intelligence*, **97**, 245–271.

Bonney, R., Cooper, C.B., Dickinson, J., Kelling, S., Phillips, T., Rosenberg, K.V. & Shirk, J. (2009) Citizen science: A developing tool for expanding science knowledge and scientific literacy. *BioScience*, **59**, 977–984.

Bouckaert, R.R. (2008) Practical bias variance decomposition. In *AI 2008: Advances in Artificial Intelligence, 21st Australasian Joint Conference on Artificial Intelligence* (eds. W. Wobcke & M. Zhang), *Lecture Notes in Computer Science*, vol. 5360, pp. 247–257. Springer-Verlag, Berlin Heidelberg, Germany.

Bradford, J.P., Kunz, C., Kohavi, R., Brunk, C. & Brodley, C.E. (1998) Pruning decision trees with misclassification costs. In *Proceedings of the 10th European Conference on Machine Learning* (eds. C. Nédellec & C. Rouveirol), *Lecture Notes in Computer Science*, vol. 1398, pp. 131–136. Springer, Berlin Heidelberg, Germany.

Breck, E. & Cardie, C. (2004) Playing the telephone game: Determining the hierarchical structure of perspective and speech expressions. In *Proceedings of the 20th International Conference on Computational Linguistics*, pp. 120–126. Association for Computational Linguistics, Morristown, NJ, USA.

Breiman, L. (1996) Bagging predictors. *Machine Learning*, **24**, 123–140.

Breiman, L. (1998) Arcing classifiers. *The Annals of Statistics*, **26**, 801–849.

Breiman, L. (2001a) Random forests. *Machine Learning*, **45**, 5–32.

Breiman, L. (2001b) Statistical modeling: The two cultures. *Statistical Science*, **16**, 199–231.

Breiman, L., Friedman, J., Stone, C.J. & Olshen, R.A. (1984) *Classification and Regression Trees*. Chapman & Hall/CRC.

Breiman, L. & Shang, N. (1996) *Born Again Trees*. Tech. rep., Department of Statistics, University of California, Berkeley. URL `ftp://ftp.stat.berkeley.edu/pub/users/breiman/BAtrees.ps`.

Bryll, R., Gutierrez-Osuna, R. & Quek, F. (2003) Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, **36**, 1291–1302.

Buchan, I., Winn, J. & Bishop, C. (2009) A unified modeling approach to data-intensive healthcare. In (Hey *et al.*, 2009), pp. 91–97.

Buciluǎ, C., Caruana, R. & Niculescu-Mizil, A. (2006) Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. T. Eliassi-Rad, L. Ungar, M. Craven & D. Gunopulos), pp. 535–541. ACM, New York, NY, USA.

Buntine, W. (1992) Learning classification trees. *Statistics and Computing*, **2**, 63–73.

Buntine, W. & Caruana, R. (1991) *Introduction to IND and Recursive Partitioning*. Tech. Rep. FIA-91-28, NASA Ames Research Center.

Buntine, W. & Niblett, T. (1992) A further comparison of splitting rules for decision-tree induction. *Machine Learning*, **8**, 75–85.

Burges, C.J.C., Ragno, R. & Le, Q.V. (2007) Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 19* (eds. B. Schölkopf, J.C. Platt & T. Hoffman), pp. 193–200. MIT Press, Cambridge, MA, USA.

Bylander, T. (2002) Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine Learning*, **48**, 287–297.

Caruana, R., Baluja, S. & Mitchell, T. (1996) Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In *Advances in Neural Information Processing Systems 8* (eds. D.S. Touretzky, M. Mozer & M.E. Hasselmo), pp. 959–965. MIT Press, Cambridge, MA, USA.

Caruana, R. & de Sa, V.R. (2003) Benefitting from the variables that variable selection discards. *Journal of Machine Learning Research*, **3**, 1245–1264.

Caruana, R., Elhawary, M., Fink, D., Hochachka, W.M., Kelling, S., Munson, A., Riedewald, M. & Sorokina, D. (2006a) Mining citizen science data to predict prevalence of wild bird species. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. T. Eliassi-Rad, L. Ungar, M. Craven & D. Gunopulos), pp. 909–915. ACM, New York, NY, USA.

Caruana, R. & Freitag, D. (1994) Greedy attribute selection. In *Proceedings of the 11th International Conference on Machine Learning* (eds. W.W. Cohen & H. Hirsh), pp. 28–36. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Caruana, R., Karampatziakis, N. & Yessenalina, A. (2008) An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning* (eds. A. McCallum & S.T. Roweis), *ACM International Conference Proceeding Series*, vol. 307, pp. 96–103. ACM Press, New York, NY, USA.

Caruana, R., Munson, A. & Niculescu-Mizil, A. (2006b) Getting the most out of ensemble selection. In *Proceedings of the 6th IEEE International Conference on Data Mining* (eds. C.W. Clifton, N. Zhong, J. Liu, B.W. Wah & X. Wu), pp. 828–833. IEEE Computer Society, Washington, DC, USA.

Caruana, R. & Niculescu-Mizil, A. (2004) Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. W. Kim, R. Kohavi, J. Gehrke & W. DuMouchel), pp. 69–78. ACM, New York, NY, USA.

Caruana, R. & Niculescu-Mizil, A. (2006) An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning* (eds. W.W. Cohen & A. Moore), *ACM International Conference Proceeding Series*, vol. 148, pp. 161–168. ACM Press, New York, NY, USA.

Caruana, R., Niculescu-Mizil, A., Crew, G. & Ksikes, A. (2004) Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning* (eds. R. Greiner & D. Schuurmans), *ACM International Conference Proceedings Series*, vol. 69. ACM Press, New York, NY, USA.

Cohen, J. (1960) A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**, 37–46.

Cohen, W.W. (1995) Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning* (eds. A. Prieditis & S.J. Russell), pp. 115–123. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Conway, C.J. & Timmermans, S.T.A. (2005) Progress toward developing field protocols for a North American marshbird monitoring program. In *Bird Conservation Implementation and Integration in the Americas: Proceedings of the Third International Partners in Flight Conference* (eds. C.J. Ralph & T.D. Rich), General Technical Report PSW-GTR-191, pp. 997–1005. U.S. Department of Agriculture Forest Service, Albany, CA.

Craven, M.W. & Shavlik, J.W. (1996) Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems 8* (eds. D.S. Touretzky, M. Mozer & M.E. Hasselmo), pp. 24–30. MIT Press, Cambridge, MA, USA.

Daelemans, W. & Hoste, V. (2002) Evaluation of machine learning methods for natural language processing tasks. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation* (eds. N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, A.M. Municio, D. Tapias & A. Zampolli), pp. 755–760. European Language Resources Association (ELRA), Paris, France.

Daelemans, W., Zavrel, J., van der Sloot, K. & van den Bosch, A. (2000) *TiMBL: Tilburg Memory Based Learner, version 3.0, Reference Guide*. ILK Technical Report 00-01, Tilburg University. Available from `http://ilk.kub.nl/~ilk/papers/ilk0001.ps.gz`.

De Falco, I., Della Cioppa, A., Iazzetta, A. & Tarantino, E. (2005) An evolutionary approach for automatically extracting intelligible classification rules. *Knowledge and Information Systems*, **7**, 179–201.

De'ath, G. & Fabricius, K.E. (2000) Classification and regression trees: A powerful yet simple technique for ecological data analysis. *Ecology*, **81**, 3178–3192.

Dietterich, T.G. (2000a) Ensemble methods in machine learning. In *Multiple Classifier Systems: 1st International Workshop* (eds. J. Kittler & F. Roli), *Lecture Notes in Computer Science*, vol. 1857, pp. 1–15. Springer, Berlin Heidelberg, Germany.

Dietterich, T.G. (2000b) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, **40**, 139–157.

Dodd, L.E. & Pepe, M.S. (2003) Partial AUC estimation and regression. *Biometrics*, **59**, 614–623.

Domingos, P. (1998) Knowledge discovery via multiple models. *Intelligent Data Analysis*, **2**, 187–202.

Domingos, P. (1999) MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. U. Fayyad, S. Chaudhuri & D. Madigan), pp. 155–164. ACM, New York, NY, USA.

Domingos, P. (2000) A unified bias-variance decomposition and its applications. In *Proceedings of the 17th International Conference on Machine Learning* (ed. P. Langley), pp. 231–238. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Donmez, P., Svore, K.M. & Burges, C.J.C. (2009) On the local optimality of LambdaRank. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (eds. J. Allan, J.A. Aslam, M. Sanderson, C. Zhai & J. Zobel), pp. 460–467. ACM Press, New York, NY, USA.

Draper, N.R. & Smith, H. (1981) *Applied Regression Analysis*. John Wiley & Sons, Inc., 2 edn.

Dudík, M., Phillips, S.J. & Schapire, R.E. (2007) Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, **8**, 1217–1260.

Efroymson, M.A. (1960) Multiple regression analysis. In *Mathematical Methods for Digital Computers* (eds. A. Ralston & H.S. Wilf), vol. 1, chap. 17, pp. 191–203. John Wiley & Sons, Inc.

Elith, J., Graham, C.H., Anderson, R.P., Dudik, M., Ferrier, S., Guisan, A., Hijmans, R.J., Huettmann, F., Leathwick, J.R., Lehmann, A., Li, J., Lohmann, L.G., Loiselle, B.A., Manion, G., Moritz, C., Nakamura, M., Nakazawa, Y., Overton, J.M.M., Peterson, A.T., Phillips, S.J., Richardson, K., Scachetti-Pereira, R., Schapire, R.E., Soberón, J., Williams, S., Wisz, M.S. & Zimmermann, N.E. (2006) Novel methods improve prediction of species' distributions from cccurrence data. *Ecography*, **29**, 129–151.

Elkan, C. (2001) The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artifical Intelligence* (ed. B. Nebel), pp. 973–978. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Esposito, F., Malerba, D. & Semeraro, G. (1997) A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 476–491.

Faanes, C.A. & Bystrak, D. (1981) The role of observer bias in the North American breeding bird survey. In *Estimating Numbers of Terrestrial Birds* (eds. C.J. Ralph & J.M. Scott), *Studies in Avian Biology*, vol. 6, pp. 353–359. Cooper Ornithological Society.

Fan, W., Stolfo, S.J., Zhang, J. & Chan, P.K. (1999) AdaCost: Misclassification cost-sensitive boosting. In *Proceedings of the 16th International Conference on Machine Learning* (eds. I. Bratko & S. Dzeroski), pp. 97–105. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Fawcett, T. (2006) An introduction to ROC analysis. *Pattern Recognition Letters*, **27**, 861–874.

Ferreira, C. (2001) Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, **13**, 87–129.

Ferrer, X., Carrascal, L.M., Gordo, O. & Pino, J. (2006) Bias in avian sampling effort due to human preferences: An analysis with Catalonian birds (1900–2002). *Ardeola*, **53**, 213–227.

Ferri, C., Flach, P. & Hernández-Orallo, J. (2002) Learning decision trees using the area under the ROC curve. In *Proceedings of the 19th International Conference on Machine Learning* (eds. C. Sammut & A.G. Hoffmann), pp. 139–146. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

Fielding, A.H. & Bell, J.F. (1997) A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environmental Conservation*, **24**, 38–49.

Fink, D., Hochachka, W.M., Zuckerberg, B., Winkler, D.W., Shaby, B., Munson, M.A., Hooker, G., Riedewald, M., Sheldon, D. & Kelling, S. (In press) Spatiotemporal exploratory models for broad-scale survey data. *Ecological Applications*.

Fitzpatrick, M.C., Preisser, E.L., Ellison, A.M. & Elkinton, J.S. (2009) Observer bias and the detection of low-density populations. *Ecological Applications*, **19**, 1673–1679.

Freund, Y., Iyer, R., Schapire, R.E. & Singer, Y. (2003) An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, **4**, 933–969.

Freund, Y. & Schapire, R.E. (1996) Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning* (ed. L. Saitta), pp. 148–156. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Friedman, J.H. (1997) On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, **1**, 55–77.

Friedman, J.H. (2001) Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, **29**, 1189–1232.

Friedman, J.H. & Popescu, B.E. (2008) Predictive learning via rule ensembles. *Annals of Applied Statistics*, **2**, 916–954.

Geman, S., Bienenstock, E. & Doursat, R. (1992) Neural networks and the bias/variance dilemma. *Neural Computation*, **4**, 1–58.

Gillam, M., Feied, C., Handler, J., Moody, E., Scheiderman, B., Plaisant, C., Smith, M. & Dickason, J. (2009) The healthcare singularity and the age of semantic medicine. In (Hey *et al.*, 2009), pp. 57–64.

Goodman, A.A. & Wong, C.G. (2009) Bringing the night sky closer: Discoveries in the data deluge. In (Hey *et al.*, 2009), pp. 39–44.

Greiner, R. & Subramanian, D. (eds.) (1994) *Relevance: Papers from the AAAI Fall Symposium*. The AAAI Press, Menlo Park, CA. Technical Report FS-94-02.

Grove, A.J. & Schuurmans, D. (1998) Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the 15th National Conference on Artificial Intelligence and 10th Conference on Innovative Applications of Artificial Intelligence* (eds. J. Mostow & C. Rich), pp. 692–699. AAAI Press, Menlo Park, CA, USA.

Guisan, A. & Thuiller, W. (2005) Predicting species distribution: Offering more than simple habitat models. *Ecology Letters*, **8**, 993–1009.

Guyon, I. & Elisseeff, A. (2003a) An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**, 1157–1182.

Guyon, I. & Elisseeff, A. (eds.) (2003b) *Special Issue of the Journal of Machine Learning Research on Variable and Feature Selection*, vol. 3. The MIT Press.

Guyon, I., Gunn, S., Ben-Hur, A. & Dror, G. (2005) Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17* (eds. L.K. Saul, Y. Weiss & L. Bottou), pp. 545–552. MIT Press, Cambridge, MA, USA.

Guyon, I., Gunn, S., Nikravesh, M. & Zadeh, L.A. (2006) *Feature Extraction: Foundations and Applications*. Springer.

Guyon, I., Weston, J., Barnhill, S. & Vapnik, V. (2002) Gene selection for cancer classification using support vector machines. *Machine Learning*, **46**, 389–422.

Hand, D., Mannila, H. & Smyth, P. (2001) *Principles of Data Mining*. The MIT Press, Cambridge, Massachusetts.

Hanley, J.A. & McNeil, B.J. (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**, 29–36.

Hanowski, J.M. & Niemi, G.J. (1995) A comparison of on- and off-road bird counts: Do you need to go off road to count birds accurately? *Journal of Field Ornithology*, **66**, 469–483.

Harrell, Jr, F.E. (2001) *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer Series in Statistics. Springer-Verlag New York, Inc., New York, NY.

Harrison, J.A., Underhill, L.G. & Barnard, P. (2008) The seminal legacy of the Southern African Bird Atlas Project. *South African Journal of Science*, **104**, 82–84.

Heckman, J.J. (1979) Sample selection bias as a specification error. *Econometrica*, **47**, 153–161.

Hegel, T.M., Cushman, S.A., Evans, J. & Huettmann, F. (2010) Current state of the art for statistical modelling of species distributions. In *Spatial Complexity, Informatics, and Wildlife Conservation* (eds. S.A. Cushman & F. Huettmann), chap. 16, pp. 273–311. Springer Japan, Tokyo. URL `http://dx.doi.org/10.1007/978-4-431-87771-4_16`.

Herbrich, R., Graepel, T. & Obermayer, K. (2000) Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers* (eds. A.J. Smola, P.J. Bartlett, B. Schölkopf & D. Schuurmans), pp. 115–132. MIT Press, Cambridge, MA, USA.

Herschtal, A. & Raskutti, B. (2004) Optimising area under the ROC curve using gradient descent. In *Proceedings of the 21st International Conference on Machine Learning* (eds. R. Greiner & D. Schuurmans), *ACM International Conference Proceedings Series*, vol. 69, pp. 49–56. ACM Press, New York, NY, USA.

Hey, T., Tansley, S. & Tolle, K. (eds.) (2009) *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, WA, USA.

Hickling, R., Roy, D.B., Hill, J.K. & Thomas, C.D. (2005) A northward shift of range margins in British Odonata. *Global Change Biology*, **11**, 502–506.

Ho, T.K. (1998) The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 832–844.

Hochachka, W.M., Caruana, R., Fink, D., Munson, A., Riedewald, M., Sorokina, D. & Kelling, S. (2007) Data-mining discovery of pattern and process in ecological systems. *Journal of Wildlife Management*, **71**, 2427–2437.

Hochachka, W.M. & Dhondt, A.A. (2000) Density-dependent decline of host abundance resulting from a new infectious disease. *Proceedings of the National Academy of Sciences of the United States of America*, **97**, 5303–5306.

Hochachka, W.M., Winter, M. & Charif, R.A. (2009) Sources of variation in singing probability of Florida grasshopper sparrows, and implications for design and analysis of auditory surveys. *The Condor*, **111**, 349–360.

Hooker, G. (2007) Generalized functional ANOVA diagnostics for high-dimensional functions of dependent variables. *Journal of Computational & Graphical Statistics*, **16**, 709–732.

Hoste, V. (2005) *Optimization Issues in Machine Learning of Coreference Resolution*. Ph.D. thesis, University of Antwerp.

Hoste, V., Hendrickx, I., Daelemans, W. & van den Bosch, A. (2002) Parameter optimization for machine learning of word sense disambiguation. *Natural Language Engineering*, **8**, 311–325.

Hunt, J.R., Baldocchi, D.D. & van Ingen, C. (2009) Redefining ecological science using data. In (Hey *et al.*, 2009), pp. 21–26.

Intrator, O. & Intrator, N. (2001) Interpreting neural-network results: A simulation study. *Computational Statistics and Data Analysis*, **37**, 373–393.

James, G. & Hastie, T. (1997) *Generalizations of the Bias/Variance Decomposition for Prediction Error*. Tech. rep., Department of Statistics, Stanford University, Stanford, CA.

Järvelin, K. & Kekäläinen, J. (2002) Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, **20**, 422–446.

Jiguet, F., Julliard, R., Thomas, C.D., Dehorter, O., Newson, S.E. & Couvet, D. (2006) Thermal range predicts bird population resilience to extreme high temperatures. *Ecology Letters*, **9**, 1321–1330.

Joachims, T. (1999) Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning* (eds. B. Schölkopf, C.J.C. Burges & A.J. Smola). MIT Press, Cambridge, USA.

Joachims, T. (2005) A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning* (eds. L. De Raedt & S. Wrobel), *ACM International Conference Proceeding Series*, vol. 119, pp. 377–384. ACM Press, New York, NY, USA.

Jobin, B., DesGranges, J.L. & Boutin, C. (1996) Comparison of BBS and intensive surveys at selected BBS stops. *Bird Populations*, **3**, 14–25.

Karalič, A. (1996) Producing more comprehensible models while retaining their performance. In *Information, Statistics, and Induction in Science: Proceedings of the Conference, ISIS '96* (eds. D.L. Dowe, K.B. Korb & J.J. Oliver), pp. 54–65. World Scientific, River Edge, NJ, USA.

Kelling, S., Hochachka, W.M., Fink, D., Riedewald, M., Caruana, R., Ballard, G. & Hooker, G. (2009) Data intensive science: A new paradigm for biodiversity studies. *BioScience*, **59**, 613–620.

Kendall, W.L., Peterjohn, B.G. & Sauer, J.R. (1996) First-time observer effects in the North American breeding bird survey. *The Auk*, **113**, 823–829.

Kéry, M., Royle, J.A., Schmid, H., Schaub, M., Volet, B., Häfliger, G. & Zbinden, N. (In press) Correcting population trend estimates from opportunistic observations for observation effort using site-occupancy species distribution modeling. *Conservation Biology*.

Kilpatrick, A.M., Chmura, A.A., Gibbons, D.W., Fleischer, R.C., Marra, P.P. & Daszak, P. (2006) Predicting the global spread of H5N1 avian influenza. *Proceedings of the National Academy of Sciences of the United States of America*, **103**, 19368–19373.

Kira, K. & Rendell, L.A. (1992) The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the 10th National Conference on Artificial Intelligence* (eds. P. Rosenbloom & P. Szolovits), pp. 129–134. AAAI Press, Menlo Park, CA, USA.

Knowles, E. (ed.) (1999) *The Oxford Dictionary of Quotations*. Oxford University Press, Oxford, UK, 5 edn.

Koenig, W.D. (2001) Spatial autocorrelation and local disappearances in wintering North American birds. *Ecology*, **82**, 2636–2644.

Kohavi, R. & John, G.H. (1997) Wrappers for feature subset selection. *Artificial Intelligence*, **97**, 273–324.

Kohavi, R. & Wolpert, D.H. (1996) Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the 13th International Conference on Machine Learning* (ed. L. Saitta). Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Kong, E.B. & Dietterich, T.G. (1995) Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning* (eds. A. Prieditis & S.J. Russell), pp. 313–321. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Kubat, M. & Matwin, S. (1997) Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning* (ed. D.H. Fisher), pp. 179–186. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Kukar, M. & Kononenko, I. (1998) Cost-sensitive learning with neural networks. In *Proceedings of the 13th European Conference on Artificial Intelligence* (ed. H. Prade), pp. 445–449. John Wiley & Sons.

Lewis, D.D. (2001) Applying support vector machines to the TREC-2001 batch filtering and routing tasks. In *The 10th Text REtrieval Conference* (eds. E.M. Voorhees & D.K. Harman), *NIST Special Publication*, vol. 550-250, pp. 286–292. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD, USA.

Lin, Y., Lee, Y. & Wahba, G. (2002) Support vector machines for classification in nonstandard situations. *Machine Learning*, **46**, 191–202.

Liu, B., Hu, M. & Hsu, W. (2000) Intuitive representation of decision trees using general rules and exceptions. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence* (eds. H.A. Kautz & B. Porter), pp. 615–620. AAAI Press, Menlo Park, CA, USA.

Lobo, J.M., Jiménez-Valverde, A. & Real, R. (2008) AUC: A misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, **17**, 145–151.

Loughrey, J. & Cunningham, P. (2005) *Using Early-Stopping to Avoid Overfitting in Wrapper-Based Feature Selection Employing Stochastic Search*. Tech. Rep. TCD-CS-2005-37, Trinity College Dublin, Department of Computer Science.

MacKenzie, D.I. (2006) Modeling the probability of resource use: The effect of, and dealing with, detecting a species imperfectly. *Journal of Wildlife Management*, **70**, 367–374.

Manel, S., Williams, H.C. & Ormerod, S.J. (2001) Evaluating presence-absence models in ecology: The need to account for prevalence. *Journal of Applied Ecology*, **38**, 921–931.

Margineantu, D.D. & Dietterich, T.G. (1997) Pruning adaptive boosting. In *Proceedings of the 14th International Conference on Machine Learning* (ed. D.H. Fisher), pp. 211–218. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Markert, B.A., Breure, A.M. & Zechmeister, H.G. (2003) Definitions, strategies, and principles for bioindication/biomonitoring of the environment. In *Bioindicators & Biomonitors: Principles, Concepts and Applications* (eds. B.A. Markert, A.M. Breure & H.G. Zechmeister), pp. 3–39. Elsevier. Trace Metals and other Contaminants in the Environment 6.

McCallum, A.K. (2002) MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

McClish, D.K. (1989) Analyzing a portion of the ROC curve. *Medical Decision Making*, **9**, 190–195.

Mingers, J. (1989a) An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, **4**, 227–243.

Mingers, J. (1989b) An empirical comparison of selection measures of decision-tree induction. *Machine Learning*, **3**, 319–342.

Mitchell, T. (1997) *Machine Learning*. McGraw-Hill.

Moisen, G.G., Freeman, E.A., Blackard, J.A., Frescino, T.S., Zimmerman, N.E. & Jr., T.C.E. (2006) Predicting tree species presence and basal area in Utah: A comparison of stochastic gradient boosting, generalized additive models, and tree-based methods. *Ecological Modelling*, **199**, 176–187.

MPQA Corpus (2002) NRRC MPQA corpus. Available from `http://nrrc.mitre.org/NRRC/Docs_Data/MPQA_04/approval_time.htm`.

Munson, M.A., Webb, K., Sheldon, D., Fink, D., Hochachka, W.M., Iliff, M., Riedewald, M., Sorokina, D., Sullivan, B., Wood, C. & Kelling, S. (2009) *The eBird Reference Dataset, Version 1.0*. Cornell Lab of Ornithology and National Audubon Society, Ithaca, NY. URL `http://www.avianknowledge.net/content/features/archive/eBird_Ref`.

Ng, V. (2004) *Improving Machine Learning Approaches to Noun Phrase Coreference Resolution*. Ph.D. thesis, Cornell University.

Ng, V. & Cardie, C. (2002) Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (eds. E. Charniak & D. Lin), pp. 104–111. Association for Computational Linguistics, Morristown, NJ, USA.

Nichols, J.D. & Williams, B.K. (2006) Monitoring for conservation. *Trends in Ecology & Evolution*, **21**, 668–673.

Niculescu-Mizil, A. & Caruana, R. (2005) Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning* (eds. L. De Raedt & S. Wrobel), *ACM International Conference Proceeding Series*, vol. 119, pp. 625–632. ACM Press, New York, NY, USA.

Niculescu-Mizil, A., Perlich, C., Swirszcz, G., Sindhwani, V., Liu, Y., Melville, P., Wang, D., Xiao, J., Hu, J., Singh, M., Xiong, W. & Zhu, Y.F. (2009) Winning the KDD Cup Orange challenge with ensemble selection. In *Proceedings of KDD-Cup 2009 Competition* (eds. G. Dror, M. Boullé, I. Guyon, V. Lemaire & D. Vogel), *JMLR Workshop and Conference Proceedings*, vol. 7, pp. 23–34. Microtome Publishing, Brookline, MA, USA.

Oates, T. & Jensen, D. (1997) The effects of training set size on decision tree complexity. In *Proceedings of the 14th International Conference on Machine Learning* (ed. D.H. Fisher), pp. 254–262. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Opitz, D.W. (1999) Feature selection for ensembles. In *Proceedings of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence* (eds. J. Hendler & D. Subramanian), pp. 379–384. AAAI Press, Menlo Park, CA, USA.

O'Sullivan, J., Langford, J., Caruana, R. & Blum, A. (2000) FeatureBoost: A meta-learning algorithm that improves model robustness. In *Proceedings of the 17th International Conference on Machine Learning* (ed. P. Langley), pp. 703–710. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T. & Brunk, C. (1994) Reducing misclassification costs. In *Proceedings of the 11th International Conference on Machine Learning* (eds. W.W. Cohen & H. Hirsh), pp. 217–225. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Pazzani, M.J. (1991) Influence of prior knowledge on concept acquisition: Experimental and computational results. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **17**, 416–432.

Pazzani, M.J. (2000) Knowledge discovery from data? *IEEE Intelligent Systems*, **15**, 10–13.

Pazzani, M.J., Mani, S. & Shankle, W.R. (1997) Beyond concise and colorful: Learning intelligible rules. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (eds. D. Heckerman, H. Mannila & D. Pregibon), pp. 235–238. AAAI Press, Menlo Park, CA, USA.

Pearce, J. & Ferrier, S. (2000) Evaluating the predictive performance of habitat models developed using logistic regression. *Ecological Modeling*, **133**, 225–245.

Pearce, J.L., Venier, L.A., Ferrier, S. & McKenney, D.W. (2002) Measuring prediction uncertainty in models of species distribution. In (Scott *et al.*, 2002), pp. 383–390.

Peterson, A.T., Papeş, M. & Soberón, J. (2008) Rethinking receiver operating characteristic analysis applications in ecological niche modeling. *Ecological Modelling*, **213**, 63–72.

Platt, J.C. (2000) Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers* (eds. A.J. Smola, P.J. Bartlett, B. Schölkopf & D. Schuurmans), pp. 61–74. MIT Press, Cambridge, MA, USA.

Quinlan, J.R. (1986) Induction of decision trees. *Machine Learning*, **1**, 81–106.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Rakotomamonjy, A. (2004) Optimizing area under ROC curve with SVMs. In *First Workshop on ROC Analysis in AI held at ECAI 2004* (eds. C. Ferri, P. Flach, J. Hernández-Orallo & N. Lachiche).

Reunanen, J. (2003) Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, **3**, 1371–1382.

Robbins, C.S. (1981a) Bird activity levels related to weather. In *Estimating Numbers of Terrestrial Birds* (eds. C.J. Ralph & J.M. Scott), *Studies in Avian Biology*, vol. 6, pp. 301–310. Cooper Ornithological Society.

Robbins, C.S. (1981b) Effect of time of day on bird activity. In *Estimating Numbers of Terrestrial Birds* (eds. C.J. Ralph & J.M. Scott), *Studies in Avian Biology*, vol. 6, pp. 275–286. Cooper Ornithological Society.

Robbins, C.S., Bystrak, D. & Geissler, P.H. (1986) *The Breeding Bird Survey: Its First Fifteen Years, 1965–1979*. Resource Publication 157, U.S. Fish and Wildlife Service.

Rosenberg, K.V. & Blancher, P.J. (2005) Setting numerical population objectives for priority landbird species. In *Bird Conservation Implementation and Integration in the Americas: Proceedings of the Third International Partners in Flight Conference* (eds. C.J. Ralph & T.D. Rich), General Technical Report PSW-GTR-191, pp. 57–67. U.S. Department of Agriculture Forest Service, Albany, CA.

Ross, S.M. (2004) *Introduction to Probability and Statistics for Engineers and Scientists*. Elsevier Academic Press, 3 edn.

Saeys, Y., Abeel, T. & Van de Peer, Y. (2008a) Robust feature selection using ensemble feature selection techniques. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Proceedings, Part II* (eds. W. Daelemans, B. Goethals & K. Morik), *Lecture Notes in Artificial Intelligence*, vol. 5212, pp. 313–325. Springer, Berlin Heidelberg, Germany.

Saeys, Y., Liu, H., naki Inza, I., Wehenkel, L. & Van de Peer, Y. (eds.) (2008b) *Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, *JMLR Workshop and Conference Proceedings*, vol. 4.

Sauer, J.R. (2000) Combining information from monitoring programs: Complications associated with indices and geographic scale. In *Strategies for Bird Conservation: The Partners in Flight Planning Process. Proceedings of the 3rd Partners in Flight Workshop* (eds. R. Bonney, D.N. Pashley, R.J. Cooper & L. Niles), RMRS-P-16, pp. 124–126. U.S. Forest Service, Rocky Mountain Research Station, Ogden, UT.

Sauer, J.R., Peterjohn, B.G. & Link, W.A. (1994) Observer differences in the North American breeding bird survey. *The Auk*, **111**, 50–62.

Schmid, H., Burkhardt, M., Keller, V., Knaus, P., Volet, B. & Zbinden, N. (2001) *Die Entwicklung der Vogelwelt in der Schweiz*. Avifauna Report Sempach 1, Annex., Swiss Ornithological Institute, Sempach, Switzerland.

Scott, J.M., Heglund, P.J., Morrison, M.L., Haufler, J.B., Raphael, M.G., Wall, W.A. & Samson, F.B. (eds.) (2002) *Predicting Species Occurrence: Issues of Accuracy and Scale*. Island Press, Washington, DC, USA.

Simons, T.R., Alldredge, M.W., Pollock, K.H. & Wettroth, J.M. (2007) Experimental analysis of the auditory detection process on avian point counts. *The Auk*, **124**, 986–999.

Skirvin, A.A. (1981) Effect of time of day and time of season on the number of observations and density estimates of breeding birds. In *Estimating Numbers of Terrestrial Birds* (eds. C.J. Ralph & J.M. Scott), *Studies in Avian Biology*, vol. 6, pp. 271–274. Cooper Ornithological Society.

Soon, W.M., Ng, H.T. & Lim, C.Y. (2001) A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, **27**, 521–544.

Stauffer, D.F. (2002) Linking populations and habitats: Where have we been? where are we going? In (Scott *et al.*, 2002), pp. 53–61.

Stohlgren, T.J., Ma, P., Kumar, S., Rocca, M., Morisette, J.T., Jarnevich, C.S. & Benson, N. (2010) Ensemble habitat mapping of invasive plant species. *Risk Analysis*, **30**, 224–235.

Subramanian, D., Greiner, R. & Pearl, J. (eds.) (1997) *Special Issue of Artificial Intelligence on Relevance*, vol. 97. Elsevier.

Sullivan, B.L., L.Wood, C., Iliff, M.J., Bonney, R.E., Fink, D. & Kelling, S. (2009) eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, **142**, 2282–2292.

Thogmartin, W.E. & Knutson, M.G. (2007) Scaling local species-habitat relations to the larger landscape with a hierarchical spatial count model. *Landscape Ecology*, **22**, 61–75.

Thomas, C.D. & Lennon, J.J. (1999) Birds extend their ranges northwards. *Nature*, **399**, 213.

Tibshirani, R. (1996a) *Bias, Variance, and Prediction Error for Classification Rules*. Tech. rep., Department of Statistics, University of Toronto, Toronto, Canada.

Tibshirani, R. (1996b) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**, 267–288.

Ting, K.M. (1998) Inducing cost-sensitive trees via instance weighting. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery* (eds. J.M. Zytkow & M. Quafafou), *Lecture Notes in Computer Science*, vol. 1510, pp. 139–147. Springer, Berlin Heidelberg, Germany.

Ting, K.M. & Zheng, Z. (1998) Boosting trees for cost-sensitive classifications. In *Proceedings of the 10th European Conference on Machine Learning* (eds. C. Nédellec & C. Rouveirol), *Lecture Notes in Computer Science*, vol. 1398, pp. 190–195. Springer, Berlin Heidelberg, Germany.

Turney, P.D. (1995) Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, **2**, 369–409.

Tuv, E. (2006) Ensemble learning. In *Feature Extraction: Foundations, and Applications* (eds. I. Guyon, S. Gunn, M. Nikravesh & L.A. Zadeh), *Studies in Fuzziness and Soft Computing*, vol. 207, chap. 7, pp. 187–204. Springer.

Tuv, E., Borisov, A., Runger, G. & Torkkola, K. (2009) Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research*, **10**, 1341–1366.

Tuv, E., Borisov, A. & Torkkola, K. (2006) Feature selection using ensemble based ranking against artificial contrasts. In *International Joint Conference on Neural Networks*, pp. 2181–2186. IEEE, Los Alamitos, CA, USA.

U.S. NABCI Committee (2000) *Bird Conservation Region Descriptions: A Supplement to the North American Bird Conservation Initiative Bird Conservation Regions Map*. Tech. rep., U.S. North American Bird Conservation Inititative Committee. Available from the Division of Bird Habitat Conservation, U.S. Fish and Wildlife Service, Arlington, VA, USA. Online at `http://www.nabci-us.org/aboutnabci/bcrdescrip.pdf`.

U.S. NABCI Monitoring Subcommittee (2007) *Opportunities for Improving Avian Monitoring*. Tech. rep., U.S. North American Bird Conservation Inititative Committee. Available from the Division of Migratory Bird Management, U.S. Fish and Wildlife Service, Arlington, VA, USA. Online at `http://www.nabci-us.org/`.

van der Putten, P. & van Someren, M. (2004) A bias-variance analysis of a real world learning problem: The CoIL challenge 2000. *Machine Learning*, **57**, 177–195.

Van Horne, B. (1983) Density as a misleading indicator of habitat quality. *Journal of Wildlife Management*, **47**, 893–901.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D. & Hirschman, L. (1995) A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference* (eds. B. Sundheim & R. Grishman), pp. 45–52. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Wallace, C.S. & Patrick, J.D. (1993) Coding decision trees. *Machine Learning*, **11**, 7–22.

Wiebe, J., Breck, E., Buckley, C., Cardie, C., Davis, P., Fraser, B., Litman, D., Pierce, D., Riloff, E., Wilson, T., Day, D. & Maybury, M. (2003) Recognizing and organizing opinions expressed in the world press. In *Papers from the AAAI Spring Symposium on New Directions in Question Answering (AAAI tech report SS-03-07)*. March 24-26, 2003. Stanford University, Palo Alto, California.

Wilson, E.O. (2003) The encyclopedia of life. *Trends in Ecology and Evolution*, **18**, 77–80.

Witten, I.H. & Frank, E. (2000) *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Francisco, CA.

Yan, L., Dodier, R., Mozer, M.C. & Wolniewicz, R. (2003) Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *Proceedings of the 20th International Conference on Machine Learning* (eds. T. Fawcett & N. Mishra), pp. 848–855. AAAI Press, Menlo Park, CA, USA.

Yang, Y. (2001) A study on thresholding strategies for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (eds. W.B. Croft, D.J. Harper, D.H. Kraft & J. Zobel), pp. 137–145. ACM Press, New York, NY, USA.

Young, B.E., Franke, I., Hernandez, P.A., Herzog, S.K., Paniagua, L., Tovar, C. & Valqui, T. (2009) Using spatial models to predict areas of endemism and gaps in the protection of Andean slope birds. *The Auk*, **126**, 554–565.

Yuan, M. & Lin, Y. (2006) Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**, 49–67.

Yue, Y., Finley, T., Radlinski, F. & Joachims, T. (2007) A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (eds. W. Kraaij, A.P. de Vries, C.L.A. Clarke, N. Fuhr & N. Kando). ACM Press, New York, NY, USA.

Zadrozny, B. & Elkan, C. (2001) Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (eds. F. Provost & R. Srikant), pp. 204–213. ACM, New York, NY, USA.

Zadrozny, B., Langford, J. & Abe, N. (2003) Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the 3rd IEEE International Conference on Data Mining* (eds. X. Wu & A. Tuzhilin), pp. 435–442. IEEE Computer Society, Washington, DC, USA.

Zou, H. & Hastie, T. (2005) Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, **67**, 301–320.