Summer 7-21-2020

Quantitatively Motivated Model Development Framework: Downstream Analysis Effects of Normalization Strategies

Jessica M. Rudd
Kennesaw State University

KENNESAW STATE UNIVERSITY

DOCTORAL DISSERTATION

Quantitatively Motivated Model Development Framework: Downstream Analysis Effects of Normalization Strategies

---

Author:

Jessica M. RUDD

Supervisor:

Dr. Herman "Gene" RAY

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

in the

College of Computing and Software Engineering

Analytics & Data Science Institute

July 21, 2020

Declaration of Authorship

I, Jessica M. RUDD, declare that this thesis titled, "Quantitatively Motivated Model Development Framework: Downstream Analysis Effects of Normalization Strategies" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Jessica Rudd

Date: 21 July 2020

"You can't accomplish anything without the possibility of failure."

Lazarus Lake - creator of the world's most difficult ultramarathon

KENNESAW STATE UNIVERSITY

Abstract

College of Computing and Software Engineering

Analytics & Data Science Institute

Doctor of Philosophy

Quantitatively Motivated Model Development Framework: Downstream Analysis Effects of Normalization Strategies

by Jessica M. RUDD

Through a review of epistemological frameworks in social sciences, history of frameworks in statistics, as well as the current state of research, we establish that there appears to be no consistent, quantitatively motivated model development framework in data science, and the downstream analysis effects of various modeling choices are not uniformly documented. Examples are provided which illustrate that analytic choices, even if justifiable and statistically valid, have a downstream analysis effect on model results. This study proposes a unified model development framework that allows researchers to make statistically motivated modeling choices within the development pipeline. Additionally, a simulation study is used to determine empirical justification of the proposed framework. This study tests the utility of the proposed framework by investigating the effects of normalization on downstream analysis results. Normalization methods are investigated by utilizing a decomposition of the empirical risk functions, measuring effects on model bias, variance, and irreducible error. Measurements of bias and variance are then applied as diagnostic procedures for model pre-processing and development within the unified framework. Findings from simulation results are included in the proposed framework and stress-tested on benchmark datasets as well as several applications.

Acknowledgements

Here is where I get to use my own voice and time to thank the incredible people in my life, hopefully without the band playing me off the stage. That also means I have no excuse if I accidentally leave someone out, but it will probably still happen so apologies in advance. First and foremost, a PhD is not a solo adventure and my husband, Adam, knows that more than anyone. He walked this path with me for the past 4 years, and took the brunt of the highs and lows, stress, mood swings, and every other emotion that comes with self-imposed academic masochism. He took it all like a champ; there is no one else in my world who lifts me up more than he does, and I'm forever grateful for the best partner in an incredible adventure.

My mom and dad, Tim and Sheryl Moore, always joke that they do not know how two 'C' students created an 'A', but 23 years of cumulative education and too many degrees later, here we are. My parents did not always understand what made me tick, but they are masters at supporting whatever each of their unique kids needed or wanted in life. From putting me on a plane by myself at age 10 to go to Space Camp to traveling all over the country for robotics competitions, they made sure my brother and I never missed an opportunity. When they moved me 1000 miles away from home to Atlanta when I was 18 for college, I did not fully understand the difficulty and sacrifice of that transition at that time. It is an amazing testament to their faith in me and my ability to face challenges (which I had yet to recognize in myself), although I am not sure they realized I would still be here 17 years later. I grew up on Long Island but my parents now live in the great state of New Hampshire. Whenever people ask me where I am from, I almost always respond, "New Hampshire", because home will always be wherever I find my mommy and daddy.

My "baby" brother, Dan, is one of my biggest supporters even though he also thinks I am a huge nerd. It only took 35 years, but I finally felt cool after I started doing some sports analytics and Dan was finally interested in my nerdiness. I knew I had made it in life when I was able to call my brother for sports stats advice. Now I just need to get a job with the Yankees and I will officially be the best sister ever. I am working on it Dan. :-)

My sister-in-law and brother-in-law, Jenn and Brandon, are the family everyone else wishes they could marry into. The time and adventures we get to enjoy together have brought so much joy to my life. Now that I have finally reached the end of this academic journey I really hope there are many more adventures to come, hopefully with me being less of an anxiety-filled mess. Thank you for your enduring love and patience.

I think the two biggest supporters of the Jessica Rudd fanclub will always be my grandparents, Marvin and Sandy Udasin (in blessed memory). Granna and Grandpop always seemed so excited seeing the personal and, particularly, academic achievements of their grandchildren. Granna was an exceptionally happy, caring person who taught me how to make the best matzoh ball soup and latkes. When I was scared during a thunderstorm, she would just tell me it was the sound of god bowling. She was a very talented afghan knitter; whenever I was having a particularly bad day trying to write this beast of a dissertation, I would sit at my desk wrapped in one of her many cozy creations. It was always like getting a hug. She left us last fall but the last time I saw her I was able to tell her that I would be the first doctor in the family very soon. Whenever I hear thunder now, I just think she must be bowling. Granpop used to joke with me when I told him I got 100 on a test, "well, why didn't you get any extra credit?" He was always very proud to share that he "graduated last at Brooklyn Tech". He was a Navy veteran and very proud American, and one of his ways of sharing that was to hand out state quarters to all the kids. It was his way of giving back and making

sure no one left his house without change in their pocket. I have a roll of Georgia State quarters sitting on my desk that I meant to hand out at my defense as a way of honoring his memory and giving something back to an incredible community who has supported me through this process. I can no longer do that in a virtual defense but will hold on to them for a time when we can all meet in person again. If you are reading this, you definitely deserve a quarter. This whole dissertation is dedicated in honor of Granna and Grandpop, and the many other family members who are no longer with us including grandfather Richard, grandmother Agnes, Uncle Rich, Aunt Jan, and my father-in-law Hollis Rudd.

I am also incredibly grateful to an amazing family of friends, most of whom are from the Atlanta running and cycling community. The Atlanta Track Club community has given me mental and physical strength, much laughter, and a little bit of insanity required to take on literal marathons as well as the metaphorical marathon that is a PhD. In my application letter to the program, I suggested that completing a PhD would be like completing a marathon or ultramarathon. I was not wrong. Very special credit is due to my fun size wonder twin, Tina, who is an incredible cheerleader, and always supported this PhD process in the best possible way; with a never-ending supply of mimosas. Stephanie and Amy round out the Charlie's Angels girl-gang that every woman deserves. Carol Gsell invited me on the best Boston Marathon weekend of all time and deserves rock star status for editing this entire dissertation on last minute notice in less than 2 days. To Michael and Brad who are always up for a Backstreet Boys concert or a trip to Six Flags. There are really so many other people I am so lucky to have in my life, I would list them all if I didn't think it would jeopardize my ability to actually graduate this year. If you are my friend from Atlanta Track Club, Bike Ride Across Georgia, or BibRave you are on the list. I owe you all a party like it's 1999 as soon as quarantine possible.

To the three incredible mentors who wrote my recommendations for entrance into this program. My biostatistics professor from Rollins School of Public Health, Paul Weiss, helped spark my love of statistics and analytics in general. His mantra, "there are people in p-values" perhaps started the obsession that resulted in this dissertation. My team lead in CDC Division of Viral Diseases, Aaron Curns, told me that he hired me to be a "Jane of all trades" and then gave me the space in my career to do just that. Atlanta Track Club coach and 2018 Olympian Amy Begley (the same Amy in the aforementioned Charlie's Angels girl gang), gave me the confidence to think of myself as an athlete and inspired me through her own perseverance on the road to the Olympics. I am honored to have her as a friend; this PhD is my own Olympics.

Of course, a PhD journey is not possible without an academic family. I am incredibly thankful to have an all-star committee of Dr. Gene Ray, Dr. Jennifer Priestley, and Dr. Lin Li. I had the pleasure of working with Dr. Li on a really fun "data for good" project for Bert's Big Adventure in 2018 and was very happy she was excited to join my dissertation committee. Her unique insight, especially after my proposal presentation, was very helpful for formulating a well-connected research problem.

I am pretty sure that Dr. Ray became my committee chair through a long series of begging and no shortage of tears, but he deserves now a long series of thanks for his invaluable input, time, and constant encouragement. One of the most difficult parts of academic research, in my opinion, is learning how to feel confident in your knowledge and ideas. Dr. Ray through his many roles as professor, adviser, dissertation chair, and a simply friendly ear when I needed it, taught me how to "make it my own" and find my academic voice. I kicked and screamed the whole way but, by the end, I can say I enjoyed it because I ended up part of an awesome academic lineage.

Dr. Jennifer Priestley, Mother of Dragons, is the most influential academic and professional cheerleader I have ever had in my life. I would not be in

this program at all had it not been for her belief in my ability to succeed in a program I was, on paper, not qualified to attend. Dr. Priestley works hard to make sure the Analytics and Data Science students have as many academic and professional opportunities as possible, and I benefited from so many of them. She wrote me countless recommendations whenever I asked, and never said no to something I was interested in pursuing. I am most thankful for her determination to make the PhD program full of talented women, as well as her ability to encourage the best work from people. I still don't think I can beat her at Galaga but, no, I'm not whinging and my heart is definitely not troubled.

There are many other faculty and staff connected with the Analytics and Data Science Institute who have been incredible resources to me over the past 4 years. In particular, Dr. Sherrill Hayes has been my fellow social sciences warrior in the Institute, and provided key encouragement for finding my own way with a dissertation topic I felt very strongly pursuing. His insights helped me articulate and connect my passions for both social science and data science. I am also immensely grateful for the many hours he allowed me to use his office during my moments of existential crises, as well as many discussions of all things nerd-related. Cara Reeve is an indispensable resource for every student and faculty within our program; I am confident we would all be quite useless without her. Dr. DeMaio, Dr. Westlund, Dr. Yang, Dr. Xie, Dr. Moazzez, Dr. Ni, Dr. Ferguson, Dr. Chowdhury, Dr. Laval, and Mr. Michael Frankel have all had incredible influence on my academic and professional development while at KSU and I would not have made it to the finish line without any one of them.

Finally, a PhD is truly not possible without the support and teamwork of awesome classmates. My fellow Cohort 2 classmates Yiyun Zhou, Lili Zhang, Yan Wang, and Shashank Hebbar are perhaps the best team of people with whom I will ever work. My "mei mei" (little sisters) Lili and Yan, are the most

brilliant women I know and never had a shortage of kindness or patience whenever I needed it the most. When I entered the program with barely enough mathematics background to survive, Bob Vanderheyden and Bogdan Gadidov provided an endless supply of notes and explanations, and Bob was always the best classroom partner. Lauren Staples is the classmate and friend every female academic deserves. Sanjoosh Akkineni often drove me crazy, but he always knows how to throw a birthday party. To all of the current and future cohorts of students: “The pursuit of a PhD is an enduring daring adventure (Lailah Gifty Akita)." I’m glad we took this adventure together.

Contents

Declaration of Authorship iii

Abstract viii

Acknowledgements ix

1 Introduction 1

1.1 An Epistemology of Data Science 1

1.2 Problem Motivation 4

1.2.1 Principles of Ethical Data Science 8

2 Literature Review 11

2.1 Introduction: The Case for a Model Development Framework 11

2.1.1 Data Collection and Sample Selection 18

2.1.2 Data Cleansing and Imputation 20

2.1.3 Feature Selection and Reduction 23

2.1.4 Feature Engineering 24

2.1.5 Normalization 27

2.1.6 Model Building 27

2.1.7 Model Training & Validation 29

2.1.8 Model Evaluation 32

2.1.9 Quantitatively Motivated Pre-Processing Framework for Models 32

2.2 Normalization Strategies in Statistics 34

2.2.1 Z-Score, "Standardization" 35

2.2.2 Min-Max Normalization 36
2.2.3 Max Absolute Value Normalization 37
2.2.4 Quantile Transformation 37
2.2.5 Quantile Normalization 38
2.3 Normalization Strategies in Machine Learning 39
2.3.1 Input Normalization 40
2.3.2 Batch Normalization 42
2.3.3 Effects of Batch Normalization 44
2.4 Effects of Model Frameworks and Normalization in Application 46
2.4.1 NCAA Basketball Event Prediction 47
2.4.2 Normalization in Genetics 52

3 Loss Function Decomposition and Model Characteristics 57
3.1 Relationship between Machine Learning and Statistical Measures of Error 58
3.2 Generalization of MSE Decomposition for Prediction 64
3.2.1 Loss Function Decomposition for Classification 67
3.2.2 0-1 Loss Decomposition for Classification 69
3.3 Pre-Processing Effects on Loss Function Decomposition 71
3.4 Quantifying Bias-Variance Tradeoff 72
3.5 Model Characteristics 72
3.5.1 Generalized Linear Models 73
Linear Regression 75
Logistic Regression 76
Poisson Regression 76
3.5.2 Decision Tree 77
3.5.3 Random Forest 79
3.5.4 Support Vector Machines (SVM) 80
3.5.5 Gradient Boosting 82

3.5.6 Neural Network 84
3.5.7 Model Summary 86

4 Methods **89**
4.1 Simulation Methods 89

5 Experimental Results **93**
5.1 Simulation Results 93
5.1.1 Binary Target 93
Logistic Regression Results 93
Decision Tree Results 94
Random Forest Results 95
Support Vector Machine (SVM) Results 95
Gradient Boosting Results 96
Neural Network Results 96
Generalized Results 97
5.1.2 Continuous Target 98
Linear Regression Results 98
Decision Tree Results 99
Random Forest Results 99
Support Vector Machine (SVM) Results 100
Gradient Boosting Results 100
Neural Network Results 101
Generalized Results 101
5.1.3 Poisson Target 103
Poisson Regression Results 103
Decision Tree Results 104
Random Forest Results 104
Support Vector Machine (SVM) Results 104
Gradient Boosting Results 105

Neural Network Results 105
Generalized Results 106
5.1.4 Framework 108
5.2 Benchmark Data Results 109
5.3 Applications 111
5.3.1 NCAA Tournament Data 111
5.3.2 Credit Risk Data 113

6 Discussion **115**
6.1 Conclusions 115
6.1.1 Limitations 118
6.1.2 Future Research 120
6.1.3 Motivating Example 120
6.1.4 Extensions of Research 122
6.1.5 Application to Ethical Principles of Data Science 123

A Tables **125**
A.0.1 Simulations 125
Bivariate Normal Data with Binary Target 125
Bivariate Normal Data with Continuous Target 126
Bivariate Normal Data with Poisson Target 127
Ranked Data with Binary Target 128
Ranked Data with Continuous Target 129
Ranked Data with Poisson Target 130
Categorical Data with Binary Target 131
Categorical Data with Continuous Target 132
Categorical Data with Poisson Target 133
Mixed Data with Binary Target 134
Mixed Data with Continuous Target 135
Mixed Data with Poisson Target 136

A.0.2 Benchmark Results 137
Wine Quality Data with Binary Target 137
Breast Cancer Data with Binary Target 138
Voting Data with Binary Target 139
Abalone Data with Binary Target 140
Arrhythmia Data with Binary Target 141
Forest Fires Data with Continuous Target 142
Solar Flares Data with Continuous Target 143
Auto MPG Data with Continuous Target 144

B Figures **145**

B.0.1 Simulations 145
Bivariate Normal Data with Binary Target 146
Bivariate Normal Data with Continuous Target 147
Bivariate Normal Data with Poisson Target 148
Ranked Data with Binary Target 149
Ranked Data with Continuous Target 150
Ranked Data with Poisson Target 151
Categorical Data with Binary Target 152
Categorical Data with Continuous Target 153
Categorical Data with Poisson Target 154
Mixed Data with Binary Target 155
Mixed Data with Continuous Target 156
Mixed Data with Poisson Target 157
B.0.2 Benchmark Data Results 158
Wine Quality Data 158
Breast Cancer Data 159
Voting Data 160
Abalone Data 161

Arrhythmia Data 162
Forest Fires Data 163
Solar Flares Data 164
Auto MPG Data 165

C Code **167**

C.1 Create Simulated Datasets 168
C.2 Example of Bias-Variance Decomposition 170

Bibliography **175**

List of Figures

2.1 Gravlee's assessment of elapsed time from immigration to birth on cephalic index (Gravlee, Bernard, and Leonard, 2003) . . . 13
2.2 Most commonly used transformations. Moving up the ladder (right) reduces negative skew. Moving down the ladder (left) reduces positive skew . . . 14
2.3 *Implications of choosing a neuroimagine pipeline*. . . . 17
2.4 From 'Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results' (Silberzahn et al., 2018) . . . 18
2.5 Example model development framework, adapted from Davenport and Harris, 2017 . . . 19
2.6 Proportions of repeated data and seen data in testing sets (Gray, Bowes, Davey, Sun, and Christianson, 2011). . . . 21
2.7 Error of estimated mean values for each imputation method (Barakat et al., 2017). . . . 23
2.8 ROC curves of wine quality (a) and arrhythmia (b) data (Zhang, Ray, Priestley, and Tan, 2019). . . . 26
2.9 Gradient descent with data on different scales (Jordan, 2019) . 28
2.10 Computational costs and validation accuracy of various models (He et al., 2019) . . . 29
2.11 Visualization of train-validate-test data splits (Shah, 2017) . . 30
2.12 k-fold cross-validation with k = 4 (Shulga, 2018) . . . 31

2.13 Proposed framework for diagnostic pairing of best models, model development strategies, and data structures 34
2.14 The data is squished due to outlier but most of the data lies within similar range for both features (**codeacademy**). 36
2.15 Min-Max Normalization fixes the distribution on the Y-axis but is still problematic on the X-axis due to the outlier ("Normalization", n.d.). 37
2.16 Machine Learning Literature Metrics 40
2.17 Example Neural Network(a) (Jordan, 2019) 42
2.18 Example Neural Network(b) (Jordan, 2019) 43
2.19 2014 NCAA tournament log losses for logistic regression models (A, B, C), stochastic gradient boosted tree model (D), neural network (E), various stacked algorithms, and two naive benchmarks. The average Kaggle log loss score was 0.58. (Yuan et al., 2015) 50
2.20 Game prediction framework as described in Hao et. al. (2018) 51
2.21 Log loss of five models in ten-fold cross validation (Hao, Kristal, and Hsu, 2018) 51
2.22 Root mean square error (RMSE) performance of normalization algorithms. (Argyropoulos et al., 2006) 55

3.1 Illustration of High Bias (Raschka, n.d.) 63
3.2 Illustration of High Variance (Raschka, n.d.) 63
3.3 Random Forest Structure (Chakure, 2020) 79
3.4 Maximum Margin Hyperplane (Kumari and Chitra, 2013) . . 81
3.5 Bagging (independent models) and Boosting (sequential models) (Grover, 2019) 83
3.6 Ensembling (Grover, 2019) 84
3.7 One hidden layer Neural Network 85

4.1 Application Datasets 91

5.1 Heatmap representing empirical risk function values as a percentage of the best performing normalization strategy for each data type, with green as best performing strategies. 109
5.2 2018 Men's Basketball March Madness bracket developed using winning model, logistic regression with raw data. 113
5.3 AUC Curves for selected model-normalization combinations tested based on model framework results 114

B.1 Bias-Variance Decomposition for Bivariate Normal Data with Binary Target 146
B.2 Bias-Variance Decomposition for Bivariate Normal Data with Continuous Target 147
B.3 Bias-Variance Decomposition for Bivariate Normal Data with Poisson Target 148
B.4 Bias-Variance Decomposition for Ranked Data with Binary Target 149
B.5 Bias-Variance Decomposition for Ranked Data with Continuous Target 150
B.6 Bias-Variance Decomposition for Ranked Data with Poisson Target 151
B.7 Bias-Variance Decomposition for Categorical Data with Binary Target 152
B.8 Bias-Variance Decomposition for Categorical Data with Continuous Target 153
B.9 Bias-Variance Decomposition for Categorical Data with Poisson Target 154
B.10 Bias-Variance Decomposition for Mixed Data with Binary Target 155
B.11 Bias-Variance Decomposition for Mixed Data with Continuous Target 156
B.12 Bias-Variance Decomposition for Mixed Data with Poisson Target 157

B.13 Bias-Variance Decomposition for Wine Quality Data with Binary Target 158
B.14 Bias-Variance Decomposition for Breast Cancer Data with Binary Target 159
B.15 Bias-Variance Decomposition for Congressional Voting Data with Binary Target 160
B.16 Bias-Variance Decomposition for Abalone Data with Binary Target 161
B.17 Bias-Variance Decomposition for Arrhythmia Data with Binary Target 162
B.18 Bias-Variance Decomposition for Forest Fires Data with Continuous Target 163
B.19 Bias-Variance Decomposition for Solar Flares Data with Continuous Target 164
B.20 Bias-Variance Decomposition for Auto MPG Data with Continuous Target 165

List of Tables

3.1 Relationships between relevant terms of loss functions (Raschka, n.d.) . . . 69
3.2 Summary of common Generalized Linear Models from Agresti 74

4.1 Benchmark Datasets and Charactertics . . . 91

5.1 Simulated model performance . . . 108

A.1 Bias-Variance Decomposition Results for Bivariate Normal Data with Binary Target . . . 125
A.2 Bias-Variance Decomposition Results for Bivariate Normal Data with Continuous Target . . . 126
A.3 Bias-Variance Decomposition Results for Bivariate Normal Data with Poisson Target . . . 127
A.4 Bias-Variance Decomposition Results for Ranked Data with Binary Target . . . 128
A.5 Bias-Variance Decomposition Results for Ranked Data with Continuous Target . . . 129
A.6 Bias-Variance Decomposition Results for Ranked Data with Poisson Target . . . 130
A.7 Bias-Variance Decomposition Results for Categorical Data with Binary Target . . . 131
A.8 Bias-Variance Decomposition Results for Categorical Data with Continuous Target . . . 132

A.9 Bias-Variance Decomposition Results for Categorical Data with Poisson Target 133
A.10 Bias-Variance Decomposition Results for Mixed Data with Binary Target 134
A.11 Bias-Variance Decomposition Results for Mixed Data with Continuous Target 135
A.12 Bias-Variance Decomposition Results for Mixed Data with Poisson Target 136
A.13 Bias-Variance Decomposition Results for Wine Quality Data with Binary Target 137
A.14 Bias-Variance Decomposition Results for Breast Cancer Data with Binary Target 138
A.15 Bias-Variance Decomposition Results for Congressional Voting Data with Binary Target 139
A.16 Bias-Variance Decomposition Results for Abalone Data with Binary Target 140
A.17 Bias-Variance Decomposition Results for Arrhythmia Data with Binary Target 141
A.18 Bias-Variance Decomposition Results for Forest Fires Data with Continuous Target 142
A.19 Bias-Variance Decomposition Results for Solar Flares Data with Continuous Target 143
A.20 Bias-Variance Decomposition Results for Auto MPG Data with Continuous Target 144

List of Abbreviations

NCAA	**N**ational **C**ollegiate **A**thletic **A**ssociation
CDF	**C**umulative **Distribution F**unction
MSE	**M**ean **S**quared **E**rror
ML	**M**achine **L**earning
CNN	**C**onvolutional **N**eural **N**etwork
PCA	**P**rinciple **C**omponent **A**nalysis
DE	**D**ifferential **E**xpression
eFDR	**e**mpirical **F**alse **D**iscovery **R**ate
GLM	**G**eneralized **L**inear **M**odels
SVM	**S**upport **V**ector **M**achine
MLE	**M**aximum **L**ikelihood **E**stimation

List of Symbols

L	**L**oss function value
M	**M**odel
P	**P**reprocessing technique
D	**D**ata structure
S	Sample
τ	Training Set
m	Sample size

Chapter 1

Introduction

1.1 An Epistemology of Data Science

"Data Science" as a term and discipline was only popularized within the last 10 to 15 years. The advent and use of the terminology is tied with the increasing scale, speed, variety, and complexity of data being generated, collected, stored, and analyzed. Depending on one's domain expertise, data science has taken on a variety of meanings ranging from data mining and database management, to machine learning and algorithm development, with a wide spectrum of potential skill sets and complexity of data in between. In the Kennesaw State University PhD program in Analytics and Data Science, data science is defined academically as the confluence of statistics, mathematics, and computer science. According to Josh Wills (who lists himself as "ex-statistician" on LinkedIn), a data scientist is a:

> Person who is better at statistics than any software engineer and better at software engineering than any statistician.

The interdisciplinary nature of data science, as well as the wide range of domains represented among self-described data science practitioners, presents several challenges towards applied research in the field. Without a unified understanding of the practice of data science, how can we quantify what will happen when we take domain specific knowledge and apply it within new

fields? Defining data science and its approach to scientific inquiry goes a long way towards understanding what it actually means to be a data scientist.

Popularized in the work by David Holpert and William Macready, the No Free Lunch (NFL) Theorem states that no single machine learning algorithm is better than all the others on all problems (Wolpert, Macready, et al., 1997). It is common to try multiple models and find one that works best for a problem. The goal in data analytics and modeling is not simply to predict the future with the best model. Instead, the goal is to use available information to inform the decisions about possible futures and outcomes. The interdisciplinary nature of data science has led to disparate opinions of the nature of inquiry within the field and how best to approach model development. For example, pioneering Microsoft computer scientist Jim Gray proposes data science as the 4th paradigm of science inquiry: that growing big data availability, new analytical methods, and the computing resources available to marry the two suggest we can analyze data without hypotheses, and let algorithms find patterns in data where science cannot (Kitchin, 2014). However, when the objective in data science becomes to find every possible association and let the data inform a narrative we would not have otherwise understood, we risk finding patterns that are not always meaningful, and correlations that are random and have no actual causal association. If you look at the clouds long enough, you will certainly see some mythical shapes.

In contrast, in Floridi's assessment of the epistemology of big data, he points out that too much data presents an epistemological problem in terms of what data to throw away, what is important, and lots of small patterns that may or may not be valuable (Floridi, 2012). He suggests the technological solution to this epistemological problem includes more and better techniques and technologies, which effectively shrink "big data" back to a manageable size. In our mind, taking the "big" problem away from "big data" brings us back to a world where statistical theory and reasoning is the key to effective

and valid pattern detection. With all this in mind, part of a researcher's potential success in data science is their ability to deduce which data is valuable, which can be dropped, and which missing data matters, making the exercise very similar to traditional data mining methods on a larger scale. As technology continues to keep pace with larger and larger amounts of data, "big data" becomes just "data". Leaning into the statistical reasoning and mathematical assumptions required to make valid deductions from this data remains unchanged, regardless of the relative size of data on a day-to-day basis. What is "big" today will not be "big" tomorrow, but bad data will always present challenges regardless of size. The art of making data of any size or structure manageable while balancing and understanding the underlying statistical requirements to properly translate the data into meaningful information is the definition of data science.

The current interdisciplinary state of data science suffers from an inconsistent approach to modeling strategies, with the complexities of big data, and potentially bad data, inadequately addressed. From a review of literature regarding the epistemology of data science, as well as the current state of research, we find that data science practitioners encounter several major challenges:

- Even big data does not exist in a vacuum
- What constitutes "big data" is not consistent across domains
- No Free Lunch (NFL) Theorem: no single algorithm is better than all others on all problems
- Choice of analytic strategy, whether statistically valid or not, has effects on the results

1.2 Problem Motivation

This dissertation was initially motivated by a model built for the NCAA® Men's Basketball tournament bracket prediction. The likelihood of predicting a perfect NCAA basketball tournament bracket with no prior knowledge is 1 in 9.2 quintillion. However, the odds increase somewhat to one in 2.4 trillion (Schwartz, 2015) if we take into account some basic knowledge of the game, such as adjusting probability based on seeding. Another study calculated the odds at one in 128 billion (Schwartz, 2015). In any case, the odds are not favorable. Current and previous work in this area show slight improvements each year to the model prediction but some suspect there to be a "ceiling" in terms of model improvement for this problem. Many models ranging from ranking methodologies to machine learning methods reach an upper limit for game prediction of approximately 75% accuracy (percent of tournament games predicted correctly), with similar results found in other sports such as soccer, American football, NCAA football, and the NBA (Shi, Moorthy, and Zimmermann, 2013). The NCAA bracket problem was added as a Kaggle Competition in 2014 and, since then, many individuals and teams have attempted to squeeze every bit of information out of the available data to "solve" the problem. The problem has attracted such interest that billionaire philanthropist Warren Buffet offered $1 billion to anyone who builds a perfect bracket. Game play data for every regular season and tournament game going back to 1985 is readily available, as well as a variety of external ratings systems and engineered statistics. In the Kaggle competition, log-loss is used for model evaluation and ranking. Winning entries since the start of the Kaggle competition have only ranged from log-loss scores between 0.46 to 0.53 with little improvement seen each year ("Google Cloud NCAA ML Competition", 2020). In fact, in Round 1 of the 2020 NCAA Kaggle tournament, before it was discontinued due to COVID-19, Kaggle moderators started

automatically removing any posted log-loss scores less than 0.2, and suggested in the competition discussion board that any scores less than 0.46 (a model as good as the winning round 2 submission from the previous 5 years) were likely suffering from data leakage and/or model overfitting. In spite of increased availability of data and improved technologies, there has been very little improvement to the bracket prediction results, regardless of model evaluation metrics used.

With limited formal knowledge of basketball, engagement in this competition required application of domain expertise from unrelated fields. However, this is a situation where naivety can be an advantage and allows the data to more "purely" inform the model, as in Jim Gray's proposal of a 4th scientific paradigm. The advantage of this approach is that it avoids over-engineering the problem. Many existing solutions to the NCAA bracket problem utilized feature engineering, often influenced by the researchers knowledge of the game, in order to reach a more "perfect" prediction. The seeds provided by the NCAA ranking system already offer good predictive power, but their predictive power diminishes in later tournament rounds as the difference in quality of the teams diminishes. This is where domain knowledge (in this case, specific knowledge of college basketball and the NCAA tournament) can provide an advantage, with savvy modelers engineering features out of existing data to more precisely distinguish between the quality of teams, especially in later rounds of the tournament. However, these models tend to miss upsets (games where a lower seeded team beats a higher seed), particularly in the first round of 64, even if they are better at predicting more closely matched games in later rounds. Other models focused specifically on predicting the upsets and engineered the models around that problem. For example, some models predict where the seeding goes wrong by generating an upset probability for each game. In games where the upset probability is greater than some threshold (selected through domain knowledge), the higher-seeded

team is predicted to lose. While there is some success with this method, there is also evidence of over-prediction of upsets which ultimately reduces the accuracy of these models to below seeding baseline (Bryan, Steinke, and Wilkins, 2006). Creating models that predict every nuance of the game resulted in overfitting and poor generalization to new data, i.e. new tournament data. Yuan et. al found through literature review and their own experimentation with a large set of features and predictive models that parsimonious feature sets and simple algorithms tend to outperform complicated models with many features (Yuan et al., 2015). Since relying on domain expertise of the game resulted in overcomplicated and, generally, overfit models, it presented an opportunity to find clues within the existing data and our assumptions about that data. Not being burdened by possible bias from too much knowledge of the game, and of the nuances and history of the tournament, was certainly an advantage but it did not mean we escaped the importance of understanding the data. Many may assume that since we have "complete" game data for every game, the data available for the NCAA problem is representative of the full problem space population and, therefore, statistical sample-based assumptions are inconsequential. However, data does not exist within a vacuum. Who decided which game statistics were important to collect and why? Do the statistics collected in 1985 still help describe the style of play and likelihood of success for teams in 2020? Even if statistics from 1985 are still relevant for game play in 2020, do they predict success now with the same amount of variability as in 1985 (i.e. consistency of the variance-covariance matrix across time)? These questions influence how that data is interpreted and the validity of its value in a predictive model.

Instead of relying on traditional pre-processing data normalization procedures, we decided to leverage public health methods and apply known genetic sequence normalization techniques to the basketball data. We used our domain knowledge from public health to notice that the vectors of team statistics,

rankings, and tournament performance can be viewed much like strings of genetic data, with each game similar to a genetic sample, and winning and losing team features viewed as expressed genes under opposing "biological" conditions (i.e. winning teams vs. losing teams). While the connection is not a perfect case in point, it provided enough inspiration to use domain-specific knowledge in a seemingly unrelated and untested application. The resulting best models we developed for the 2019 tournament utilized this genetic pre-processing method, but it was statistically unclear why these methods worked best. Reviewing available research regarding data normalization in the genetic sequencing space, normalization is often study-specific or based on assumptions of consistency (homeogeneity) within and between sample data distributions, resulting in potential issues of study replication and inconsistent results (Evans, Hardin, and Stoebel, 2017). For example, the assumption that the statistical distributions of samples in a dataset are the same, and that observed variation around data is consistent and the result of technical variation, may result in the removal of valuable information and true biological affects if inappropriate normalization methods are applied. In this context there appears to be no consistent, quantitatively motivated model development framework, and the downstream analysis effects of the various modeling choices are not uniformly documented.

This study proposes a unified model development framework. The proposed framework is then tested by quantifying the downstream analysis effects of data pre-processing choices by utilizing a decomposition of the loss functions, measuring effects on model bias, variance, and irreducible error/random noise. In this way, measurements of bias and variance can be efficiently applied as diagnostic procedures for model pre-processing and development. Even when considering "big data" problems, focusing on the effects of model development choices on the resulting downstream analysis and data structures can allow for a more consistent approach to the model development framework,

once again turning "big data" into just "data". Applying bias-variance decomposition to a variety of data distributions and model types can lead towards an improved understanding of quantitative variations within model development methods as well as comparing results consistently between methods. Understanding of statistical bias and variance can be used to diagnose problems with machine learning bias and develop methods for reducing bias and variance in algorithms. For example, this bias-variance trade-off does not always behave as expected under distributional assumptions. Even with the availability of more advanced models, such as neural networks, simple models still often perform well, or even better than more complex models, in experiments (Yuan et al., 2015). Generally, while more complex models result in decreased bias, they tend to increase variance and, therefore, do not generalize well to new data (Singh, 2018). However, it has been found that ensemble models, although complex, often outperform single models and this seems contradictory to the trade-off between simplicity and accuracy. In this case, decomposition of bias-variance for ensembles led to the understanding that while increased complexity for a single model often increases variance, averaging multiple models will often (but not always) lead to decreased variance (Domingos, 2000). The goal, therefore, of understanding the effects on bias-variance decomposition of various model development choices is to build a quantitatively motivated model development decision framework that results in improved performance and consistent, reproducible results across data types and domains.

1.2.1 Principles of Ethical Data Science

It is also important to point out that a unified model development framework lends itself to an improved ethical application of data science. For example, students at the Analytics and Data Science Institute at Kennesaw State University developed Principles of Ethical Data Science. These principles include:

1. Principle of responsible data collection and sourcing
2. Principle of protection
3. Principle of transparency and reproducibility
4. Principle of foresight
5. Principle of competence

With a unified framework for predictive model development within the data science community, there can be a consistent, quantitatively motivated treatment of analytic problems which, at minimum, leads to improved transparency of the transformation of data into products (principle of transparency and reproducibility). Even when more complex and advanced algorithms are considered unexplainable "black boxes", a well documented, explainable, and reproducible model development framework leads to improved confidence that the data used in these complex algorithms is well understood, statistically valid, appropriate for the modeling goal, and treated consistently by practitioners from diverse domains. Rather than risk "garbage-in, garbage-out", improvements to the efficiency of data consumption can be made.

The rest of this dissertation is organized as follows. Chapter 2 reviews relevant work presenting the case for a model development framework in data science, proposes a unified framework, and addresses details of one specific aspect of the proposed framework, i.e. data normalization. Chapter 3 presents details of the risk function decompositions to be used for an empirical study of the downstream analysis effects of normalization and selected model choices. Additional details are provided for a selection of models to be empirically tested including generalized linear models, decision tree, random forest, support vector machines, gradient boosting regression, and neural networks. Chapter 4 provides details on the bootstrap simulation methods to be used for quantifying the downstream analysis effects of the normalization methods and selected

models. Chapter 5 documents the empirical results of the simulation study as well as results on a selection of benchmark datasets from the UCI Machine Learning Library. Findings from simulation results and benchmark datasets are used as diagnostic decision points within the proposed framework, and the resulting framework is then used on updated analyses of two existing studies, i.e NCAA bracket prediction and commercial credit risk scoring. Finally, in Chapter 6 there is a discussion of the generalized results, limitations of the study, suggestions for future research, and framework application within the context of suggested Principles of Ethical Data Science.

Chapter 2

Literature Review

2.1 Introduction: The Case for a Model Development Framework

In fields of social science, theoretical frameworks are common, and often required, elements of academic research. When the goal of scientific inquiry is to formulate and test theories to explain or predict phenomena, and challenge or extend existing knowledge, theoretical frameworks provide the structure for supporting these theories ("Organizing Your Social Sciences Research Paper: Theoretical Framework", 2020). "A strong theoretical framework gives research a sound scientific basis, demonstrates understanding of existing knowledge on the topic, and allows the reader to evaluate the guiding assumptions of the research. It gives research direction, allowing one to convincingly interpret, explain and generalize from the research findings (Vinz, 2019)." Frameworks provide ("Organizing Your Social Sciences Research Paper: Theoretical Framework", 2020):

- Means by which new research data can be interpreted and coded for future use
- Response to new problems that have no previously identified solutions strategy

- Means for identifying and defining research problems
- Means for prescribing or evaluating solutions to research problems
- Ways of discerning certain facts among the accumulated knowledge that are important and which facts are not
- Means of giving old data new interpretations and new meaning
- Means by which to identify important new issues and prescribe the most critical research questions that need to be answered to maximize understanding of the issue
- Means of providing members of a professional discipline with a common language and a frame of reference for defining the boundaries of their profession
- Means to guide and inform research so that it can, in turn, guide research efforts and improve professional practice

As an example, study frameworks in biological anthropology have allowed reanalysis of existing data and scientific findings under modern analytic methods. Clarence Gravlee (2003) uses modern analysis frameworks in biological anthropology to reanalyze a classic dataset, Franz Boas's landmark study 'Changes in Bodily Form of Descendants of Immigrants.' The classic study was a crucial piece in turning the tide against early-20th century scientific racism by concluding that "cranial form changed in response to environmental influences within a single generation of European immigrants to the United States," providing analytic contradiction to the existing ideas of biological determinism. The study was highly controversial at the time in its conclusions and Boas responded to critics of the study by publishing the raw dataset. Gravlee, nearly a century later, used modern analytical methods to reevaluate Boas's hypotheses regarding human biological plasticity (the effects of the environment on human bodily

form). Using the original data and hypotheses under modern analytic frameworks, Gravlee's study supports Boas' original findings with more precise understanding of the influence of environment and lifestyle on cranial form, specifically allowing for more granularity in the effects of time elapsed from immigration (Figure 2.1).

TABLE 5. Regression of age- and sex-standardized cephalic index on time elapsed and mother's stature, by immigrant group.

	Bohemian	Central Italian	Hebrew	Hungarian and Slovak	Polish	Scotch	Sicilian
N	862	786	1,065	169	128	82	479
Time Elapsed							
β	−.099	−.068	−.141	−.025	−.132	−.118	.098
SE	.032	.036	.026	.090	.099	.083	.042
p	.004	.056	.000	.752	.138	.309	.032
Mother's Stature							
β	.049	−.031	.007	.003	−.130	.157	−.008
SE	.004	.006	.005	.014	.016	.013	.007
p	.147	.379	.828	.972	.143	.177	.868
Adjusted R^2	.009	.003	.018	−.011	.016	.004	.006
Model *p*	.007	.117	.000	.950	.132	.321	.097

Note: Square-root transformation of time elapsed; β = standardized regression coefficient.

FIGURE 2.1: Gravlee's assessment of elapsed time from immigration to birth on cephalic index (Gravlee, Bernard, and Leonard, 2003)

Extending the use of frameworks to the field of data science, one can find a precedence for such frameworks in the history of statistics. In his work, 'The Teaching of Statistics' (1940), Harold Hotelling argued for the establishment of statistics as its own field. Hotelling argued that statistics has a large enough body of techniques that it should be taught in its own department. However, since those techniques are embedded in mathematical knowledge, Hotelling suggested that statistics departments should be affiliated with math departments. At that point in time in the field of mathematical statistics, statistics was so embedded in mathematics and the technical aspects of mathematics that many academics were writing papers that focused on derivations and proofs, rather than applications. Hotelling's paper became a key point in the history of statistics as its own field, separate from mathematics, and the creation of academic departments of statistics. John Tukey further distinguished statistics as its own field, and set the precedent for frameworks of inquiry in the field, in 'The Future of Data Analysis' (1962). Tukey established

"data analysis" as the term for what applied statisticians do and distinguished it from theoretical statistics. Tukey reasoned that statistics should contribute more to data analysis by applying existing strategies to solve data analysis problems, rather than provide new techniques. He discussed existing, non-novel statistical techniques that had not yet been applied to data analysis. Examples where mathematical statistics methods had not yet been applied to complex data analysis problems included: dealing with "spotty" data (missingness, outliers, etc), multiple response data, data generated by stochastic process, data heterogeneous in precision, etc. An example of one of Tukey's frameworks for statistical analysis is Tukey's Ladder of Transformations, Figure 2.2, which gives an orderly way of re-expressing skewed data using power transformations to approximate normal distributions and reveal linear relationships (Tukey, 1977).

Tukey's Ladder of Transformations

λ	-2	-1	-1/2	0	1/2	1	2
y	$\frac{1}{x^2}$	$\frac{1}{x}$	$\frac{1}{\sqrt{x}}$	$\log x$	$\sqrt{x}$	x	x^2

FIGURE 2.2: Most commonly used transformations. Moving up the ladder (right) reduces negative skew. Moving down the ladder (left) reduces positive skew

In the ensuing decades since Tukey established "data analysis" as a practice of applied statistics, growing big data availability, new analytical methods, and the computing resources available to marry the two (as in Jim Gray's 4th Paradigm of Science Inquiry (Kitchin, 2014)) have led to the need for updated frameworks to address these more complex problems. Data science, and predictive modeling in general, lacks a unifying theoretical framework for consistently formulating and testing theories. As with Kitchin (Kitchin, 2014), there are others that claim availability of big data presents a unique

mode of conducting analytical studies outside of the requirements of traditional sample-based statistics. Mayer-Schonberger and Cukier (2013) claim that big data science research is a new mode of science with its own epistemology and norms where "data can become so big as to encompass all the available data on a phenomenon of interest". This claim contributes to debates over whether sampling is unnecessary since researchers can have "all" the data, resulting in the reduction, or even elimination, of bias and error. However, Leonelli (Leonelli, 2014) counters these points from the view of biological sciences, specifically pointing out that very few data within experimental biology are formatted in ways that make them compatible with datasets from other sources. Biological data are also difficult to integrate into a new research context due to lack of standardization in their format and production techniques, as well as an absence of stable reference materials. Curators of biological databases have strong influence on the data journey due to the need to select, format, and classify data to comply with multiple standards and the needs of diverse communities involved in biological research. As a result, big data collections in biology are still small, biased samples. Finally, Lowrie (2017) makes the case that the inescapably applied nature of data science still warrants its own framework for gaining new knowledge in the field, arguing there are two ends of the spectrum of inquiry in data science: exploration of concrete domains of applied tasks, and the inquiry into the nature and functioning of the algorithms used in these explorations. Many areas of research, such as simulation studies, development of new computing architectures, and data security research, fall along this spectrum and are often tied together. For example, data scientists maintain relationships with real world applied tasks to ensure continued access to novel data that then leads to new problem spaces to explore and solve.

Popularized in the work by David Holpert and William Macready, the No Free Lunch (NFL) Theorem states that no single machine learning algorithm

is better than all the others on all problems (Wolpert, Macready, et al., 1997). Other researchers have tried multiple models to find one that works best for a problem. In fact, studies by Carp (2012a,2012b) illustrate effects on research findings in functional MRI (fMRI) studies due to variations in analytic strategy, with increased model flexibility leading to higher rates of false positive results. Wagenmakers, et. al. (2012) point out that many studies in psychology do not commit to an analysis method before seeing the data, with some researchers fine-tuning their analysis to the data, proposing that researchers "preregister their studies and indicate in advance the analyses they intend to conduct" in order to be considered as "confirmatory" research, rather than as "exploratory". A study published in May 2020 expands on Carp's findings, noting that fMRI analyses conducted on the same data by seventy different laboratories produced a wide range of results (Botvinik-Nezer et al., 2020). This particular study highlighted the fact that fMRI analysis requires several stages of pre-processing and analysis to determine which areas of the brain show activity. They found that the choice of pre-processing pipeline led to widely varied results. Among the seventy study teams, no two teams selected the same pipeline. Figure 2.3 illustrates the potential implications of varying pipeline choices in neuroimaging.

Perhaps the most illustrative lack of research consistency is the study by Silberzahn, et. al. (2018) which recruited 29 independent research teams with 61 analysts to address the question, "Are soccer referees more likely to give red cards to dark-skin-toned players than to light-skin-toned players?" The research teams represented 13 countries, a variety of disciplines, and a range of expertise and academic degrees. Using the same dataset and research question, the 29 teams utilized 29 unique analytical modeling approaches resulting in 21 unique combinations of covariates, 20 teams with significant positive results, and odds ratios ranging from 0.89 to 2.93, as in Figure 2.4. To say the least, analytic choices, even if justifiable and statistically valid,

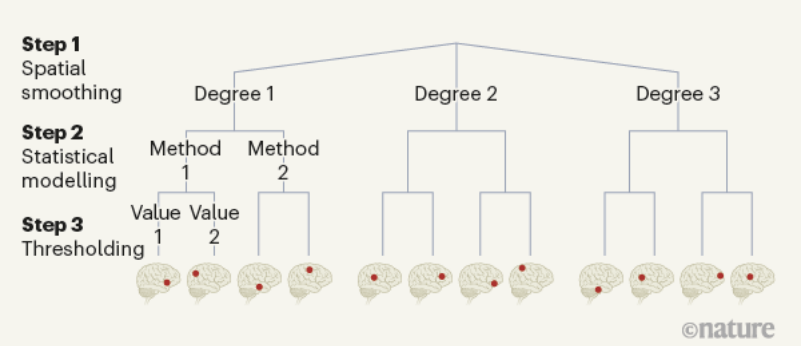


FIGURE 2.3: Researchers process neuroimaging data using a wide variety of pipelines, which can produce varying results. In this simplified example, the pipeline has three steps: spatial smoothing of the images to reduce noise, which in this example is done to three different degrees; statistical modelling, which in this example can be performed in one of two ways; and 'thresholding' of statistical tests associated with these models to determine the level at which neuronal activity in each brain region is deemed to be significant, which in this example is set to two different values. Making different choices for each step leads to a different end point — the red dots represent how activation moves throughout the brain depending on which pipeline is used (Botvinik-Nezer et al., 2020).

have a downstream effect on model results. There appears to be no unified, quantitatively motivated model development framework for making these analytic choices.

The goal of this research is to propose a unified model development framework that allows researchers to make statistically motivated variable preparation and model selection choices within the development pipeline. The model development framework can be generally divided into three phases: data discovery, variable preparation, and modeling. Within each of these phases there are steps in the model development that encompass a wide range of data management, data mining, and data analysis techniques, including data ingestion, sample selection, data cleaning and imputation, feature reduction, feature engineering, normalization, model development, and model validation. An example of a model development framework is illustrated below in Figure 2.5 and is adapted from Davenport and Harris, 2017. We propose that analyzing the downstream effects of modeling approaches within each of these steps

Table 3. Analytic Approaches and Results for Each Team

Team	Distribution	Treatment of nonindependence	Number of covariates	Analytic approach	OR
1	Linear	Clustered standard errors	7	Ordinary least squares regression with robust standard errors, logistic regression	1.18 [0.95, 1.41]
6	Linear	Clustered standard errors	6	Linear probability model	1.28 [0.77, 2.13]
14	Linear	Clustered standard errors	6	Weighted least squares regression with clustered standard errors	1.21 [0.97, 1.46]
4	Linear	None	3	Spearman correlation	1.21 [1.20, 1.21]
11	Linear	None	4	Multiple linear regression	1.25 [1.05, 1.49]
10	Linear	Variance component	3	Multilevel regression and logistic regression	1.03 [1.01, 1.05]
2	Logistic	Clustered standard errors	6	Linear probability model, logistic regression	1.34 [1.10, 1.63]
30	Logistic	Clustered standard errors	3	Clustered robust binomial logistic regression	1.28 [1.04, 1.57]
31	Logistic	Clustered standard errors	6	Logistic regression	1.12 [0.88, 1.43]
32	Logistic	Clustered standard errors	1	Generalized linear models for binary data	1.39 [1.10, 1.75]
8	Logistic	None	0	Negative binomial regression with a log link	1.39 [1.17, 1.65]
15	Logistic	None	1	Hierarchical log-linear modeling	1.02 [1.00, 1.03]
3	Logistic	Variance component	2	Multilevel logistic regression using Bayesian inference	1.31 [1.09, 1.57]
5	Logistic	Variance component	0	Generalized linear mixed models	1.38 [1.10, 1.75]
9	Logistic	Variance component	2	Generalized linear mixed-effects models with a logit link	1.48 [1.20, 1.84]
17	Logistic	Variance component	2	Bayesian logistic regression	0.96 [0.77, 1.18]
18	Logistic	Variance component	2	Hierarchical Bayes model	1.10 [0.98, 1.27]
23	Logistic	Variance component	2	Mixed-model logistic regression	1.31 [1.10, 1.56]
24	Logistic	Variance component	3	Multilevel logistic regression	1.38 [1.11, 1.72]
25	Logistic	Variance component	4	Multilevel logistic binomial regression	1.42 [1.19, 1.71]
28	Logistic	Variance component	2	Mixed-effects logistic regression	1.38 [1.12, 1.71]
21	Miscellaneous	Clustered standard errors	3	Tobit regression	2.88 [1.03, 11.47]
7	Miscellaneous	None	0	Dirichlet-process Bayesian clustering	1.71 [1.70, 1.72]
12	Poisson	Fixed effect	2	Zero-inflated Poisson regression	0.89 [0.49, 1.60]
27	Poisson	None	1	Poisson regression	2.93 [0.11, 78.66]
13	Poisson	Variance component	1	Poisson multilevel modeling	1.41 [1.13, 1.75]
16	Poisson	Variance component	2	Hierarchical Poisson regression	1.32 [1.06, 1.63]
20	Poisson	Variance component	1	Cross-classified multilevel negative binomial model	1.40 [1.15, 1.71]
26	Poisson	Variance component	6	Hierarchical generalized linear modeling with Poisson sampling	1.30 [1.08, 1.56]

Note: Values in brackets are 95% confidence intervals (CIs). Each team's observed effect size is presented in this table as an odds ratio, but some of the teams reported effect sizes in other units that were converted to odds ratios. Those originally reported effect sizes are as follows—Team 4: Cohen's d = 0.10, 95% CI = [0.10, 0.10]; Team 11: Cohen's d = 0.12, 95% CI = [0.03, 0.22]; Team 10: β = 0.01, 95% CI = [0.00, 0.01]; Team 21: β = 0.28, 95% CI = [0.01, 0.56]; Team 12: incidental risk ratio (IRR) = 0.89, 95% CI = [0.49, 1.60]; Team 27: IRR = 2.93, 95% CI = [0.11, 78.66]; Team 13: IRR = 1.41, 95% CI = [1.13, 1.75]; Team 16: IRR = 1.32, 95% CI = [1.06, 1.63]; Team 20: IRR = 1.40, 95% CI = [1.15, 1.71]; Team 26: IRR = 1.30, 95% CI = [1.08, 1.56].

FIGURE 2.4: From 'Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results' (Silberzahn et al., 2018)

should be an important goal of the data science community in order to make better informed, statistically motivated modeling choices in the future. The downstream effects of some techniques within the model development process have been addressed with prior research to varying degrees but more work in this area is required to complete a more unified model development framework.

2.1.1 Data Collection and Sample Selection

Data collection and sample selection are important steps in the model development process that can affect model interpretation and evaluation. A particularly relevant area where data collection has led to biased results has been found in facial recognition models where bias occurs due to a lack of diversity in the training data (Vincent, 2018). For time series models, we might consider the

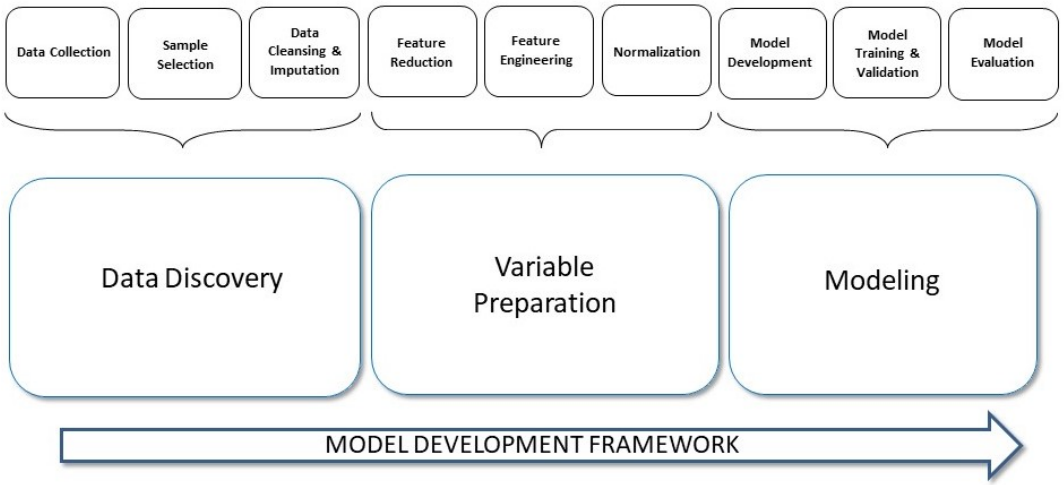


FIGURE 2.5: Example model development framework, adapted from Davenport and Harris, 2017

effects of time intervals for data collection on prediction results. Accuracy of traffic prediction models relies heavily on the data collection time interval, but it has been common for studies to arbitrarily select time intervals without consideration of the impact on prediction results. Different applications of traffic prediction models require different time intervals. For example, traffic speed prediction calculated using large time intervals has limited model capacity and can result in missed details from a dynamic traffic operation status (the model is too simple, potentially biased); the prediction results are unable to be applied in a traffic control strategy. If the time interval is too small, the calculation is time consuming and the traffic speed prediction results are unstable (potentially overfit with high model variance). A study by Song, et. al. (Song, Guo, Wu, and Ma, 2019) compared accuracy of a best performing model under different time collection intervals from one minute to 30 minutes. Results found significant improvement in traffic speed prediction with increasing time collection interval from one to 10 minutes, with slower improvement from 10 to 30 minutes, indicating that an increase of the data collection time interval leads to decreased volatility of measured traffic speed, resulting in more stable and more predictable time series, but a balance of model complexity and performance is an important consideration.

Considering the effects of sample size is also important. Sample size impacts the precision of our estimates. Generally, the more data we have, the more information we have about the true population, therefore decreasing our uncertainty and increasing precision (Littler, 2018). In gene expression research, it was found that increasing sample size led to increased prediction accuracy and stability of results. However, performance improvement varied between patient subgroups, suggesting that sample size selection should also take required study aims into account. For example, increasing sample sizes of specific gene-signature sub-groups may be a better strategy for prediction of heterogeneous diseases such as breast cancer (Kim, 2009).

2.1.2 Data Cleansing and Imputation

Data cleansing and imputation methods include filtering, listwise deletion, mean and median imputation, regression imputation, multiple imputation, and others. Data cleansing is a form of data management where data is removed or updated to account for incomplete, incorrect, improperly formatted, duplicated, or irrelevant data ("What is Data Cleansing?: Experian Business", n.d.). Incorrect or inconsistent data can lead to false conclusions if not properly addressed, but using incorrect methods of cleansing and imputation can also lead to loss of important information. Big data, with its already high volume, velocity, and variety, presents additional challenges and importance for data cleansing including resources required to parse the data, potential issues merging data from multiple sources, and the challenges of complex data structures. A study by Gray, Bowes, et. al (2011) found that misuse and improper cleaning of the publicly available NASA Metrics Data Program data sets can lead to erroneous findings. The 13 public use data sets are available for researchers to test software defect prediction experiments. The researchers found through an extensive data cleansing process that each of

the datasets had between six and 90 percent less of their original data after cleansing, with much of the data lost due to duplication. One of the major issues of duplicate data in a prediction experiment is the potential inclusion of data in the test data set that was already seen in the training data set. This is an issue known as "data leakage" and results in overfit models that do not generalize well to real-world data. In this case, the study found via 10-fold cross validation of training and test data splits, that the average "proportion of seen data points in the testing sets is larger than the proportion of repeated data points in total (Gray et al., 2011), as seen in Figure 2.6.

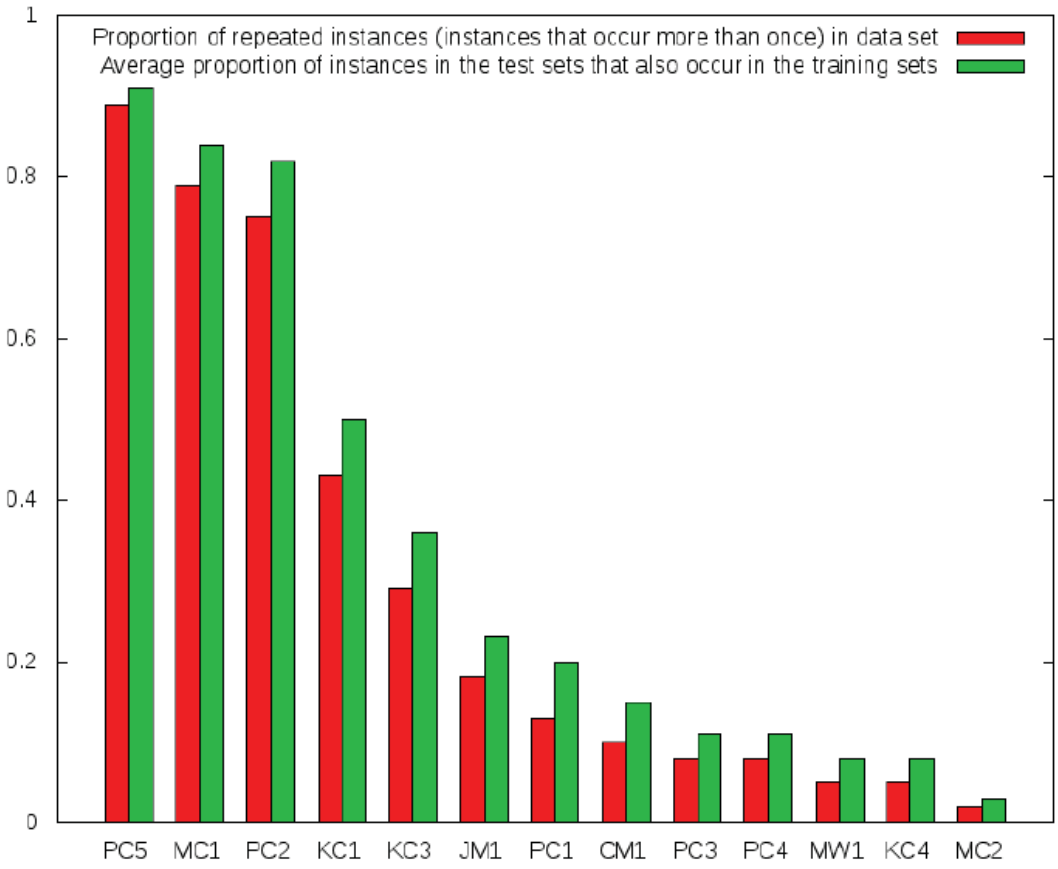


FIGURE 2.6: Proportions of repeated data and seen data in testing sets (Gray, Bowes, Davey, Sun, and Christianson, 2011).

Imputation is a technique used to replace missing data. Since many statistical analysis algorithms rely on complete data, missing data can introduce bias or affect the generalization of the results by eliminating any cases with missing

data (listwise deletion). In addition, eliminating cases with missing data results in smaller effective datasets available for modeling, and the potential for unreliable results when the number of features starts to outweigh the number of cases. For example, in a regression problem fit using least squares, when the number of observations approaches the number of features, variance of the least squares model tends to increase resulting in overfitting and poor generalization to new data. When the number of features is larger than the number of observations, there is no longer a unique estimate of the least squares model due to infinite variance, resulting in a model that will not converge.

Imputation generally falls into three categories: univariate (replacing values with the mean or median of that feature), bivariate (data stratified by a feature both related to the outcome of interest and associated with missing data, and the missing data imputed within each group), and multivariate (missing values obtained by regression of non-missing features). Using mean imputation has the advantage of maintaining the same sample mean and sample size for the imputed feature, but it can minimize any correlations involving the feature since the imputed value will have no relationship with any other measure features. Mean imputation is also not recommended for features with skewed distributions or outliers as it gives more weight to values in the skewed or outlier direction. In this case, median imputation is a better technique although it still suffers from attenuated correlations. Bivariate or stratified imputation is a good way to deal with data that is not missing at random, i.e. missingness in one feature is correlated to values in another feature. Multivariate (regression-based) imputation solves some of the problems of univariate imputation by using a regression model to predict missing values based on values of other features. However, single regression imputation fits the missing values perfectly without any error or variance in their estimates, which can result in overfitting. Multiple imputation is used to account for

imputation-related noise by averaging imputed values from multiple imputed datasets. Barakat, et. al. (Barakat et al., 2017) studied the effect of imputation methods on a training dataset for a lung cancer prediction problem and compared to prediction using a smaller, complete data set. Results showed that even when proportion of records with missing data is very high, imputation led to models with higher accuracy than those trained using smaller, complete records datasets, with multiple imputation performing best at imputing values, even when 100% of the cases contain missing data, Figure 2.7.

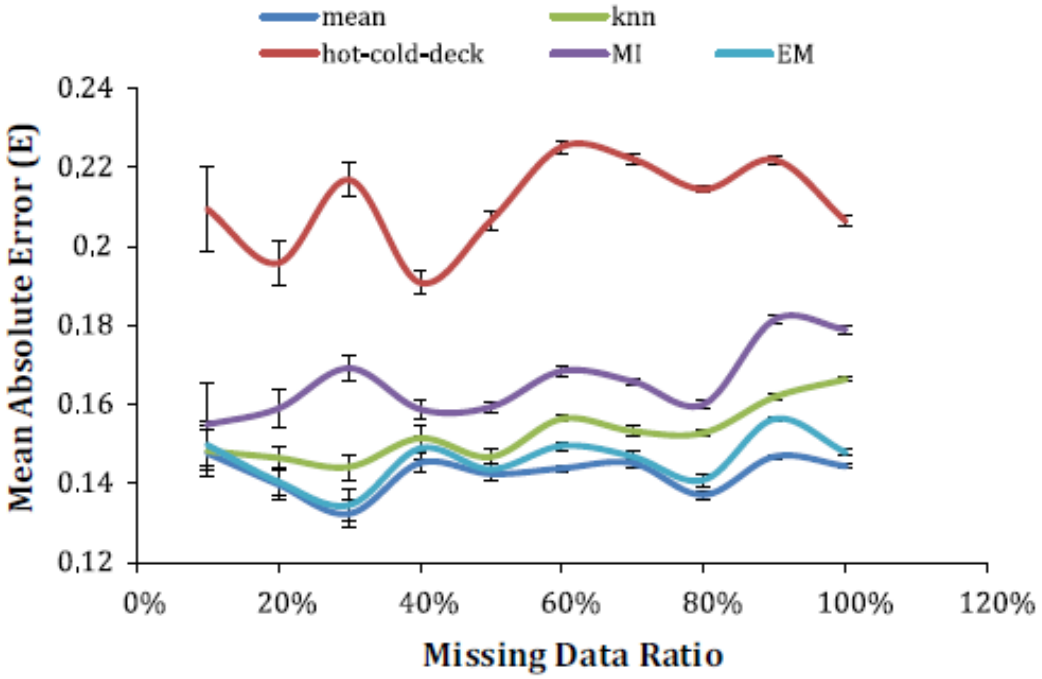


FIGURE 2.7: Error of estimated mean values for each imputation method (Barakat et al., 2017).

2.1.3 Feature Selection and Reduction

In the model development phase of feature selection and feature reduction, data scientists have a myriad of choices ranging from clustering algorithms, principle component analysis (PCA), correlation coefficients, etc. Limiting the number of features in a model helps avoid issues of multicollinearity, but the analyst also needs to balance the resulting reduction in proportion

of variance of the target feature explained when data is removed from a model. Feature selection and reduction techniques aim to balance model parsimony (simplicity) with model accuracy. Feature selection techniques reduce features in a model by selecting the most important ones, while feature reduction techniques reduce features by creating new, combined features from the originals. A 2019 study on software defect prediction was one of the first to consider the effects of several feature reduction techniques on software defect prediction (Kondo, Bezemer, Kamei, Hassan, and Mizuno, 2019). They studied the impact of eight feature reduction techniques on prediction performance and variance among five supervised and five unsupervised learning models and compared results with the best-performing feature selection techniques found in prior work. They found, for example, that

> "studied correlation and consistency-based feature selection techniques result in the best-performing supervised defect prediction models, while feature reduction techniques using neural network-based techniques (restricted Boltzmann machine and autoencoder) result in the best-performing unsupervised defect prediction models."

2.1.4 Feature Engineering

The next step of a model development framework considers feature engineering methods, such as variable discretization, transformation, and aggregation across multiple data sources. These processes create new features from raw data, often for the purpose of converting data into analysis and machine learning-ready formats, such as mapping text to numeric features. Feature engineering is also used when domain knowledge is applied to extract more relevant information from the raw data sources. Feature engineering is used to construct and select explanatory variables for model training to improve predictive power. In addition, using feature engineering aims to improve the

explanatory quality and information density of the features, allowing for the flexibility to use less complex, more parsimonious models which are more computationally efficient and easier to interpret.

In NCAA Basketball tournament prediction, researchers have made use of domain knowledge, feature selection, and feature engineering to improve prediction results, with research teams such as Dutta et. al. (2017) and Ford and Fodor (2018) focusing on scenarios of predicting upsets (the lower seeded team beats the favorite). Ford and Fodor noted with over 70 million March Madness brackets each year, the difference between winning and losing in a bracket pool is often determined by correctly predicting upsets. On average there are about 6 upsets each year in the first round (round of 64). Their proposed method combined independent datasets with historical seed data to predict matchups which are similar to historical upsets. They argue that the Rating Pecentage Index (RPI), one of the main metrics (although supposedly not the only metric) used by the NCAA to select and seed the teams for the tournament, is outdated as it does not incorporate factors such as margin of victory, location of games played, and game tempo. Many researchers have developed models to attempt to outperform this seed-based bracket selection method. While seeds are strongly predictive in early rounds, their predictive power weakens as the tournament progresses due to smaller differences between team strength, so additional selection methods become more important to consider in later rounds when finding subtle predictive features can strengthen a model over simple seed-selection. The NCAA recognizes four researchers at the forefront of analytics: Ken Pomeroy, Jeff Sagarin, Ben Alamar, and Kevin Pauga. These researchers have each created their own ratings systems that many bracket modelers now incorporate in their models because they include factors not included in the NCAA seeding-based system. Ford and Fodor combined the 4 z-normalized ratings to create an overall score for each team and found some inconsistencies between the external

ratings versus the NCAA seeded ratings, finding some teams they felt were unfairly seeded. This method also allowed the authors to compare absolute and relative strength of teams on each seed line. The method presented in their paper combines the absolute and relative strength of teams with historical upset data to make per round predictions on which teams will be involved in an upset.

Recent work by Zhang, et. al. (2019) considers the effect of discretization on classification of imbalanced data, and points out that "most standard statistics and machine learning models are heavily biased towards the majority class (i.e. non-events) and severely misclassify the minority class (i.e. events), caused by their assumptions of equal target class distribution and maximizing overall accuracy." This study illustrates the effects of variable discretization on classification of imbalanced data across several domains, comparing four discretization methods (distance, quantile, Gini, optimal binning). Variable discretization resulted in model improvement across all the test domains for measures of accuracy in comparison to models with the original variables, for example Figure 2.8.

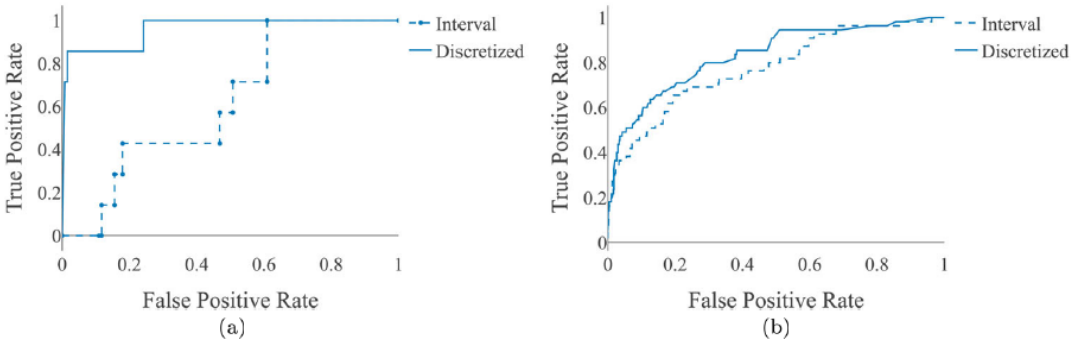


FIGURE 2.8: ROC curves of wine quality (a) and arrhythmia (b) data (Zhang, Ray, Priestley, and Tan, 2019).

2.1.5 Normalization

In this context we consider normalization to include data scaling techniques such as normalization and standardization. In traditional statistical modeling, such as logistic regression, scaling data by normalization and standardization is important because variables with a large difference in ranges can result in an ill-conditioned design matrix and difficulty reaching model convergence, resulting in slower processing times and unstable estimates. In addition, variables measured on different scales will not contribute equally to the analysis, with potential bias towards variables with much larger scales than others in the dataset (Lakshmanan, 2019).

Scaling data is equally as important in machine learning techniques such as neural networks because we optimize these models through the process of gradient descent (Jordan, 2019), where the variable inputs are put through a series of linear combinations and non-linear activations, and the optimal value of the loss function (global minimum) is found by taking steps in the direction of the steepest loss function descent. If the variables are on different scales the shape of the loss function will result in an emphasis on certain variable gradients, and difficulty in finding the true function minimum as shown in Figure 2.9. Normalizing the variables to be on the same scale allows gradient descent to optimize more quickly and converge if a global minimum exists. Additionally, scaling variables between -1 and 1 helps avoid computational issues with floating point number precision (Altman, Gill, and McDonald, 2004).

2.1.6 Model Building

In the model building phase, it is first important to select the appropriate model given the data structures. For example, data that is assumed to have some linear deterministic relationship with a continuous target of interest can

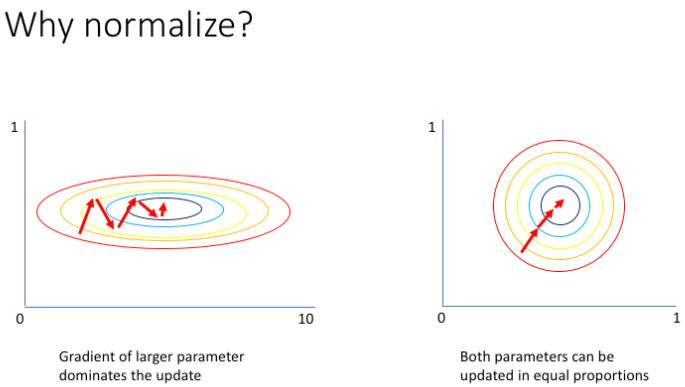


FIGURE 2.9: Gradient descent with data on different scales (Jordan, 2019)

be modeled using linear regression. Generalizations of ordinary linear model regression, the generalized linear model (GLM), can then be extended to data with non-normal response variables, such as when the target is binary or multinomial. Logistic regression and multinomial logistic regression, respectively, are appropriate in these cases. When the target data is a measure of counts, the GLM can use the Poisson or negative binomial distributions, depending on the variance in the data. These same ideas can be extended to more complex models, such as decision trees, random forests, and neural networks where the optimized loss functions are learned with estimates of a continuous value or estimates of a predicted probability for continuous and binary targets, respectively.

Once the appropriate model(s) are selected, models can be tuned to optimize prediction results using a variety of model building choices including regularization, batch processing, and other optimization techniques such as dropout and early stopping rules. While much progress in image recognition classification has focused on advancements in model architectures, gains in model accuracy have also been influenced by updates to training procedures, such as data augmentations and optimization methods. However, much of the research mentions only the implementation details or source code of such refinements,

without focus on their quantitative value. A late 2018 study by He et al. (He et al., 2019). evaluated the prediction effects of a dozen convolutional neural network training refinements, including batch size, learning rate, and weight decay. Results found that several of these refinements improve model accuracy with only minor modifications to model architecture and loss function, and that stacking all the refinements led to significantly higher accuracy, for example, raising ResNet-50's top-1 validation accuracy from 75.3% to 79.29% on ImageNet, Figure 2.10.

Model	FLOPs	top-1	top-5
ResNet-50 [9]	3.9 G	75.3	92.2
ResNeXt-50 [27]	4.2 G	77.8	-
SE-ResNet-50 [12]	3.9 G	76.71	93.38
SE-ResNeXt-50 [12]	4.3 G	78.90	94.51
DenseNet-201 [13]	4.3 G	77.42	93.66
ResNet-50 + tricks (ours)	4.3 G	**79.29**	**94.63**

FIGURE 2.10: Computational costs and validation accuracy of various models (He et al., 2019)

2.1.7 Model Training & Validation

In order to determine the best performing model architecture and attributes in the model building phase, it is necessary to train, tune, and test the models using some split of the available data into training, validation, and test datasets (See Figure 2.11). The training dataset is used to fit the model parameters (i.e. feature coefficients in regression, weights in neural network) to result in an optimal value of the predetermined loss function (i.e. minimum MSE for linear regression). Depending on the complexity of the model, there may be hyperparameters to be tuned while fitting the model on the training data. A separate validation dataset is used to fine-tune the model hyperparameters without introducing bias into the training dataset by using it for repeated model evaluation. The test dataset is then used as a hold-out sample of unseen data, representative of the population of interest, i.e. gold standard

data to evaluate the model. The test dataset is also used to compare performance of multiple models.

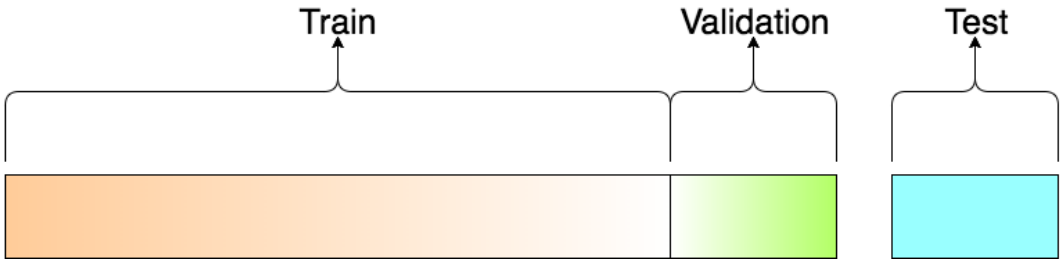


FIGURE 2.11: Visualization of train-validate-test data splits (Shah, 2017)

One of the main drawbacks of using the train-validate-test split to train, tune, and test models is that it requires enough samples to effectively split into the datasets. In a dataset with only 100 samples, using an 80-10-10 split means there is only 10 samples left in the test set. Performance results in this case would be due to chance, especially if it is a multi-class problem. A simple model with few or no hyperparameters can be tuned using a small validation set or no validation set at all, but this still leaves too few samples for testing. Cross-validation or bootstrapping are data splitting methods that allow training and testing using all the available data. Cross-validation uses multiple splits of the training data into training and validation sets (or training and test sets) where the model can be iteratively trained and tuned (or trained and tested where validation is unnecessary) using the complete data. For example, k-fold cross-validation splits the data into k parts, where the model is trained on k - 1 parts, tested on the remaining part, and then the process is repeated until each split has been used as the test dataset (See Figure 2.12). Advantages of cross-validation include: ability to use all available data for training and testing without introducing bias or overfitting, averaging model performance over multiple iterations of model training and testing (can assess whether model has consistent results), ability to split and train data based on dependent or grouped data, ability to tune hyperparameters

in complex model architectures with limited data by cross-validating on 3 folds (each fold is used as train, test, and split at least once). Xu and Goodacre (2018) tested multiple splitting methods on simulated datasets of varying sizes and found that performance results varied greatly between validation and test datasets across all methods when used on small simulated datasets, with consistency of results improving as sample size increased. In this study, they tested a selection of cross-validation, bootstrapping, and systematic partitioning (selecting most representative samples based on distribution of data and using remaining samples for validation) techniques, pointing out that, while "all these methods have been routinely reported in the literature and despite their popularity, most people chose a method with which they have familiarity."

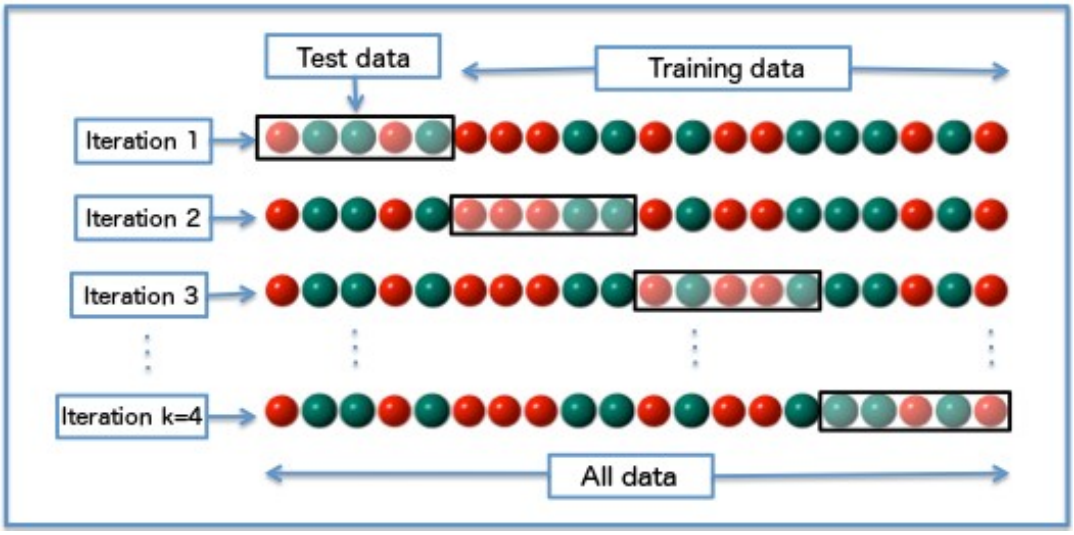


FIGURE 2.12: k-fold cross-validation with k = 4 (Shulga, 2018)

Bootstrapping is a similar method to cross-validation where, instead of folds, a re-sampling method is used for estimating the model parameters. Random samples with replacement are selected from the available data for training and then validated or tested on the out-of-bag samples. This process is then repeated many times with average results "estimating the generalization performance of the model (Xu and Goodacre, 2018)".

2.1.8 Model Evaluation

At the end of our proposed model development pipeline we have a wide selection of model evaluation measures to choose from including: accuracy, value of loss/cost functions, receiver operating characteristics (ROC) curve, sensitivity, specificity, as well as methods for calculating these measures, such as cross-validation. Even though accuracy is still the most commonly used metric to evaluate performance of machine learning models, other methods are often more appropriate given certain goals of prediction or aspects of the data, such as class imbalance. In 2019, Dinga, et. al. (Dinga, Penninx, Veltman, Schmaal, and Marquand, 2019) evaluated several families of performance measures on pattern recognition models and found that accuracy had the "worst performance with respect to statistical power, detecting model improvement, selecting informative features, and reliability of results." They also pointed out that accuracy should be avoided in models where relative cost of positive or negative prediction is relevant, such as in a clinical setting where costs of false positives and false negatives are not equivalent.

2.1.9 Quantitatively Motivated Pre-Processing Framework for Models

The current state of data science suffers from an inconsistent approach to modeling strategies, with the complexities of big data, and potentially bad data, inadequately addressed. Summarizing the previous sections:

- Even big data does not exist in a vacuum
- What constitutes "big data" is not consistent across domains
- No Free Lunch (NFL) Theorem: no single algorithm is better than all others on all problems

- Choice of analytic strategy, whether statistically valid or not, has effects on the results

Understanding and further analyzing the downstream effects of the model development strategies mentioned in the preceding sections is an important step towards a unified model development framework. Quantifying the analysis effects of these strategies in a unified framework will provide a diagnostic illustration of where researchers can expect to find improvements in their model results. The general idea of the proposed framework illustrated below is that researchers can select analysis methods based on the understanding that model results are a function of the selected model, the selected model development strategies, and the characteristics of the data:

$$L = f(M, [p_1, ...p_i], D) \tag{2.1}$$

where L is the loss function value, M is the selected model, $p_1...p_i$ is the list of model development strategies used, and D is the data structure. Figure 2.13 is the proposed model development framework. While we encourage further depth of research into each of the aforementioned steps to complete a more robust statistically motivated model development framework, the rest of this dissertation will focus on illustrating the use of the proposed framework through the quantitative effects of normalization methods. In this context, this literature review will now focus on the study of the downstream analysis effects of the various normalization methods and further development of the motivated pre-processing frameworks for the model development process. The review will touch on the normalization strategies and methods, the downstream analysis effects of normalization strategies, and the influence normalization methods have in conducting a study.

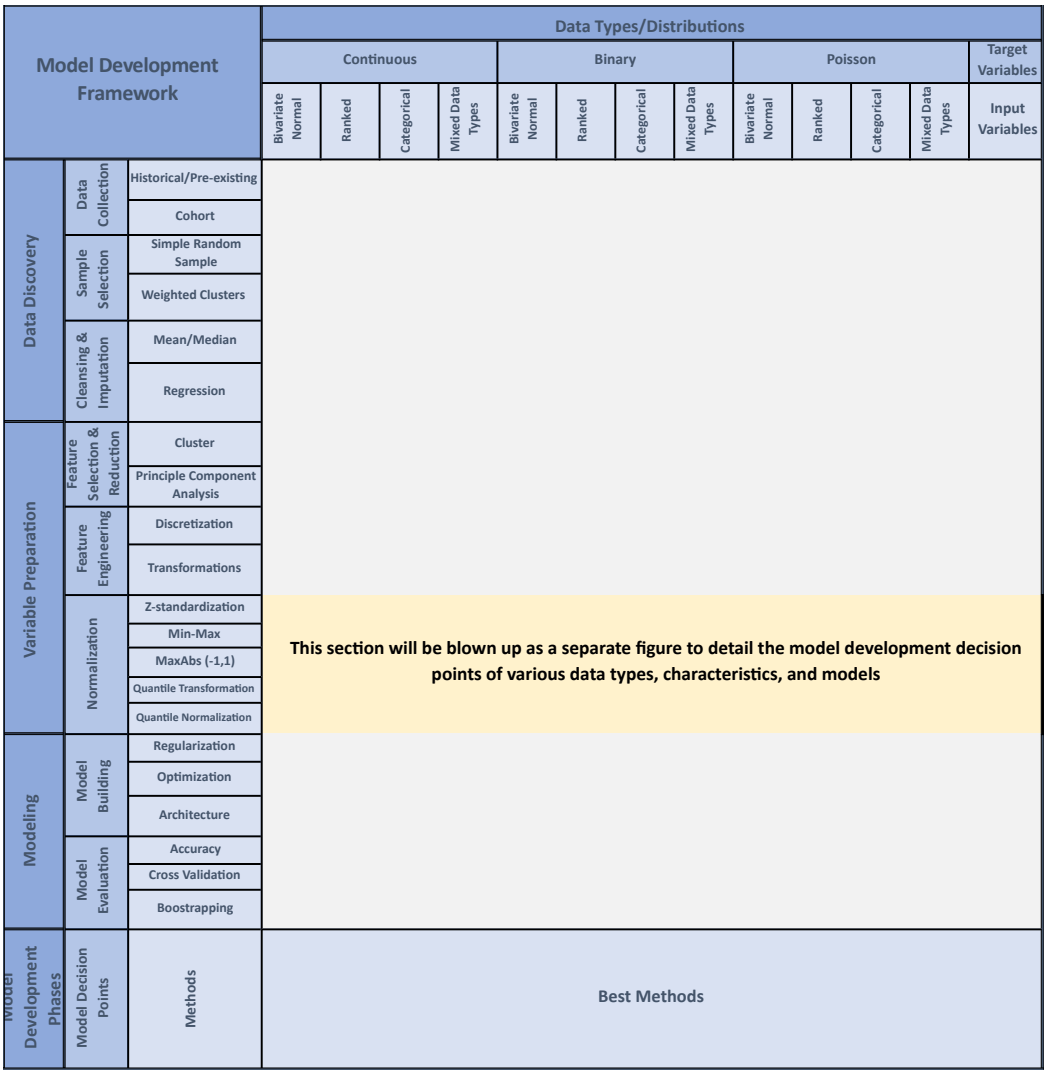


FIGURE 2.13: Proposed framework for diagnostic pairing of best models, model development strategies, and data structures

2.2 Normalization Strategies in Statistics

Normalization as it relates to data scaling and standardization in statistics was initially discussed in Section 2.1.5. Data can be scaled so that features measured on different scales can be compared on a common scale. Probability distributions of features can also be adjusted to be in alignment with each other. Another method is to shift and scale the features (standardization), which removes the units of measure. The primary goal of normalization is to scale each data point in a way that gives equal weight to the features to be used in developing a model. Five normalization methods are discussed

below, representing a range of methods used across several domains.

2.2.1 Z-Score, "Standardization"

Standardization is a method that shifts and scales the data to be centered around 0 with a standard deviation of 1:

$$\frac{x_i - mean(x)}{stdev(x)} \tag{2.2}$$

Characteristics of this method include:

- Assumes data is normally distributed within each feature
- Centers distribution around 0, with standard deviation 1
- If data has outliers, scales most of the data to a small interval
- Does not produce normalized features with the exact same scale

Even if the data has outliers, Z-score normalization will scale most of the non-outlier data to be in a similar range between all features, assuming the data is normally distributed, as in Figure 2.14.

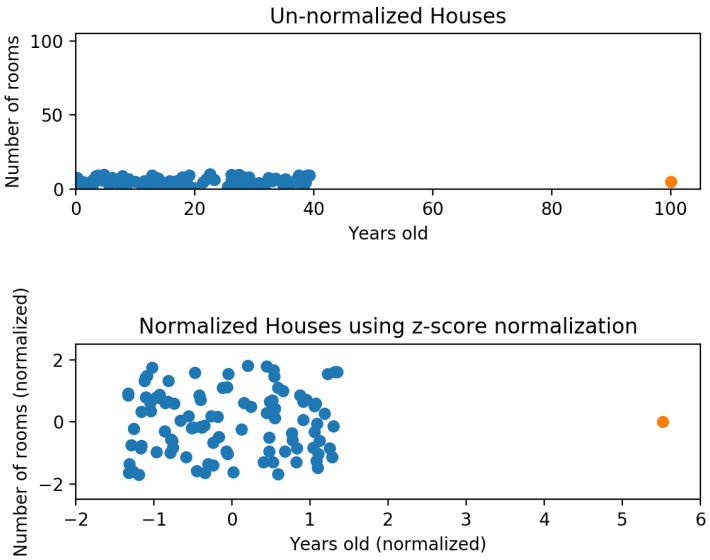


FIGURE 2.14: The data is squished due to outlier but most of the data lies within similar range for both features (**codeacademy**).

2.2.2 Min-Max Normalization

For each feature, the minimum value of that feature is transformed to a 0, the maximum value is transformed to a 1, and every other value lies between 0 and 1:

$$\frac{x_i - min(x)}{max(x) - min(x)} \tag{2.3}$$

Advantages of this method include:

- Scales data between 0 and 1; guarantees all features have exact same scale
- Preserves shape of original distribution
- Preserves 0 entries in sparse data
- Least disruptive to information in original data

However, this method does not reduce the importance of outliers, so skewed results can still exist after normalizing if outliers exist, as in Figure 2.15.

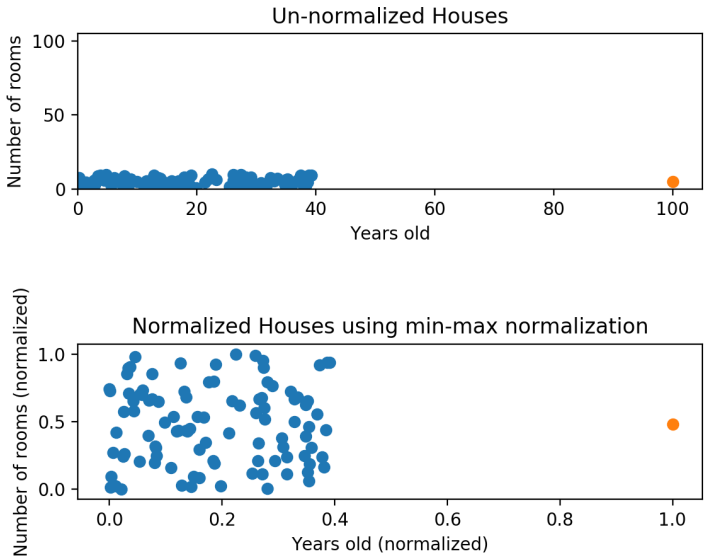


FIGURE 2.15: Min-Max Normalization fixes the distribution on the Y-axis but is still problematic on the X-axis due to the outlier ("Normalization", n.d.).

2.2.3 Max Absolute Value Normalization

This method scales feature by its maximum absolute value so that the maximum absolute value of each feature will be 1. This sets the distribution of each feature between -1 and 1.

$$\frac{x_i - mean(x)}{max(abs(x_i - mean(x)))} \tag{2.4}$$

Characteristics of this method include:

- Good for data with positive and negative values
- Preserves 0 entries in sparse data
- Similar sensitivity to outliers as in min-max normalization

2.2.4 Quantile Transformation

Quantile transformation transforms each feature independently to follow a uniform or normal distribution. This is a non-linear transformation that uses

the estimated value of the cumulative distribution function (CDF) to map a features original values to a uniform or normal distribution:

1. Calculate empirical ranks, using percentile function
2. Modify the ranking through interpolation
3. Map to a Normal distribution by inverting the CDF, and clipping bounds at the extreme values so they don't go to infinity

Characteristics of this method include:

- Tends to spread out the most frequent values of a given feature
- Smooths out unusual distributions
- Less sensitive to outliers as other scaling methods
- Distorts the linear correlations between variables measured at the same scale, but variables measured at different scales are more directly comparable
- For a Normal transformation, the median of the feature becomes the mean, centered at 0

2.2.5 Quantile Normalization

Quantile normalization is a method most notably used in genetics to normalize within samples, rather than within features as in the previously described methods. In genetic sequencing, data is often normalized based on the assumption of consistent within and between sample distributions, with observed variation around these distributions assumed to be the result of technical noise. Samples are normalized to the same distribution as each other or to a reference gene sample (Evans et al., 2017).

1. Given n arrays of length p, form X of dimension $p \times n$ where each array is a column;

2. Sort each column of X to give X_{sort};

3. Take the means across rows of X_{sort} and assign this mean to each element in the row to get X'_{sort};

4. Get $X_{normalized}$ by rearranging each column of X'_{sort} to have the same ordering as original X

Characteristics of this method include:

- Makes 2 or more distributions identical in statistical properties
- Does not preserve original data distributions

2.3 Normalization Strategies in Machine Learning

Within the context of machine learning research, normalization as it relates to data pre-processing, i.e. data scaling, is rarely mentioned in detail, if at all, and only then glossed over as a minor methodological step. In August 2019, Google Scholar released the *Google Scholar Metrics Ranking*, which ranked the most highly-cited papers published between 2014 and 2018 with all citations indexed as of July 2019. Table 2.16 details some of the most highly-cited machine learning papers from the most influential journals on the list, and illustrates the lack of detail found regarding data pre-processing.

Literature Metrics - Highly Cited Machine Learning Papers					
Author (s), title	Journal	Year	Citations	Purpose	Normalization Discussed?
He, K., Zhang, X., Ren, S., & Sun, J. *Deep residual learning for image recognition.*	Proceedings of the IEEE conference on computer vision and pattern recognition	2016	43,318	Architecture for improved training of deep layer neural networks (NN)	Details of convolutional filters and downsampling used in NN architecture included in implentation, with reasoning following conventions from previous studies. No further discussion of input scaling
LeCun, Y., Bengio, Y., & Hinton, G. *Deep Learning.*	Nature	2015	24,435	Introduction of deep learning as a machine learning method to learn complex data relationships	Only indication of importance of normalizing data is in quote: "As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure."
C. Szegedy *et al ., Going deeper with convolutions.*	IEEE Conference on Computer Vision and Pattern Recognition (CVPR)	2015	20,647	Introduction of state-of-the-art CNN architecture 'Inception'	Details of architecture, convolutions, filters, pooling layers but no discussion of pre-processing except cropping images "224×224 in the RGB color space with zero mean", which was based on previous architectures
J. Long, E. Shelhamer and T. Darrell, *Fully convolutional networks for semantic segmentation .*	IEEE Conference on Computer Vision and Pattern Recognition (CVPR)	2015	15,207	Use of a fully convolutional network for semantic segmentation task - pixelwise prediction	Discussion of architecture convolution/pooling/filter layers, training on image patches, augmenting data with noise; no comment on input scaling

FIGURE 2.16: Machine Learning Literature Metrics

2.3.1 Input Normalization

In supervised machine learning, neural networks map input data from a training set to predicted outputs. The weights of the model, similar to estimated coefficients in a regression model, are initialized to random values close to zero and updated using an algorithmic process known as gradient descent by iteratively checking estimates of error on the training dataset, as shown in equation 2.5.

$$\theta_i := \theta_i + \Delta\theta_i$$
$$\text{where} \tag{2.5}$$
$$\delta\theta_i = -\alpha\frac{\partial J(\theta)}{\partial\theta_i}$$

where $\Delta\theta_i$ is the step the algorithm takes along the gradient, with the learning rate, α, controlling the step size. Since the gradient descent algorithm

relies on an initialization of small (usually random) weights and updated calculations of error between predictions and expected values, the size of inputs and outputs used to train the model are crucial to the implementation of the gradient descent optimization problem (Brownlee, 2019). Gradient descent updates the model weights in the steepest direction which minimizes the difference between predicted and true values (found by partial differentiation). By minimizing the weights in the steepest direction, the gradient descent algorithm aims to find the parameters that minimize the loss function, with Mean Square Error (MSE), 0-1 loss, and log-loss as commonly used loss functions. For example, assume a simple neural network with input feature x_1, with values from 0 to 1, and input feature x_2, with values from 0 to 10 (as seen previously in Figure 2.9). Since the network learns the optimal combination of these inputs through a series of linear combinations and nonlinear activations, the weights (parameters) associated with each input will be on different scales, with the gradient of the larger input dominating the gradient descent updates. This leads to a loss function topology that is difficult to navigate, and slow or unstable for the model to learn due to the presence of multiple points of local minimum in the loss function and potential saddle points. The result is a model learning process that is slow, has higher variance in results, or possibly no model convergence at all (Jordan, 2019). Scaling the inputs before training the model allows for faster, more consistent optimization of the weights in the input layer. If the input variable has a normal distribution it can be standardized (mean 0, unit variance); otherwise it can be normalized (min-max scaling between 0 and 1). It is also recommended to consider scaling the output variable so that the scale of the output variable matches that of the activation function used in the output layer of the neural network, although it is best to choose an activation function that best suits the distribution of the output data (Brownlee, 2019). Scaling input data between -1 to 1 helps avoid computation accuracy issues associated with floating point number

precision, i.e. computers operating on really small or really large values (Altman et al., 2004).

2.3.2 Batch Normalization

While limited formal research has been dedicated towards neural network input normalization, there has been extensive work in recent years extending the idea of input normalization to additional hidden layers of neural networks through a technique known as "batch normalization", where batch normalization specifically utilizes the idea of data scaling to improve model performance. Other optimization techniques used with neural network architecture include dropout, weight decay, and early stopping. The central assumption of batch normalization is that if normalizing inputs to the network allows for improved learning of the parameters in the first layer, then extending this idea to the hidden layers will allow for improved learning of the parameters in those layers as well (See Figures 2.17 and 2.18).

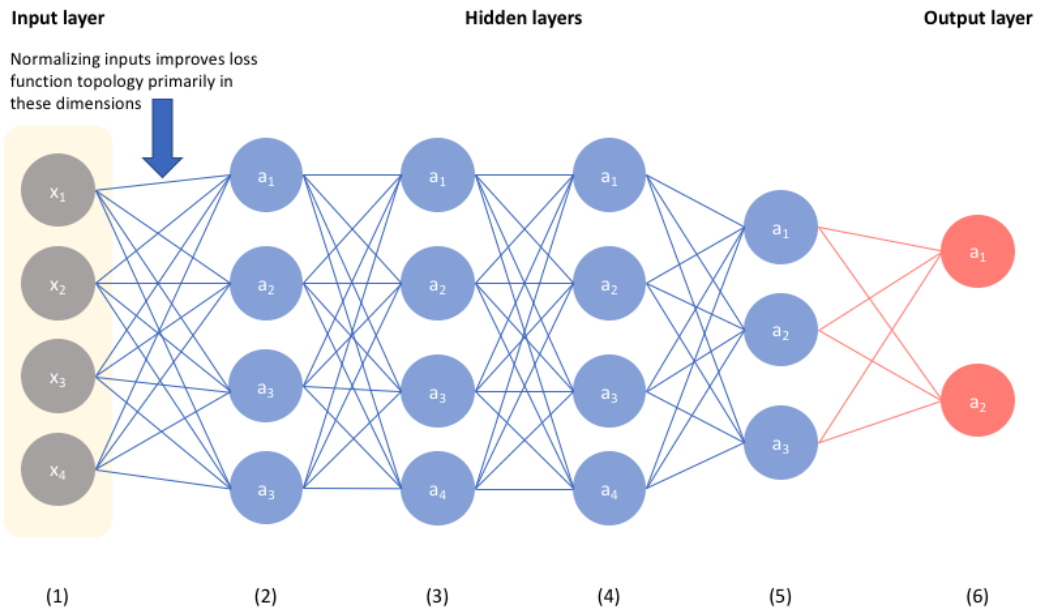


FIGURE 2.17: Example Neural Network(a) (Jordan, 2019)

"By ensuring the activations of each layer are normalized, we can simplify the overall loss function topology. This is especially

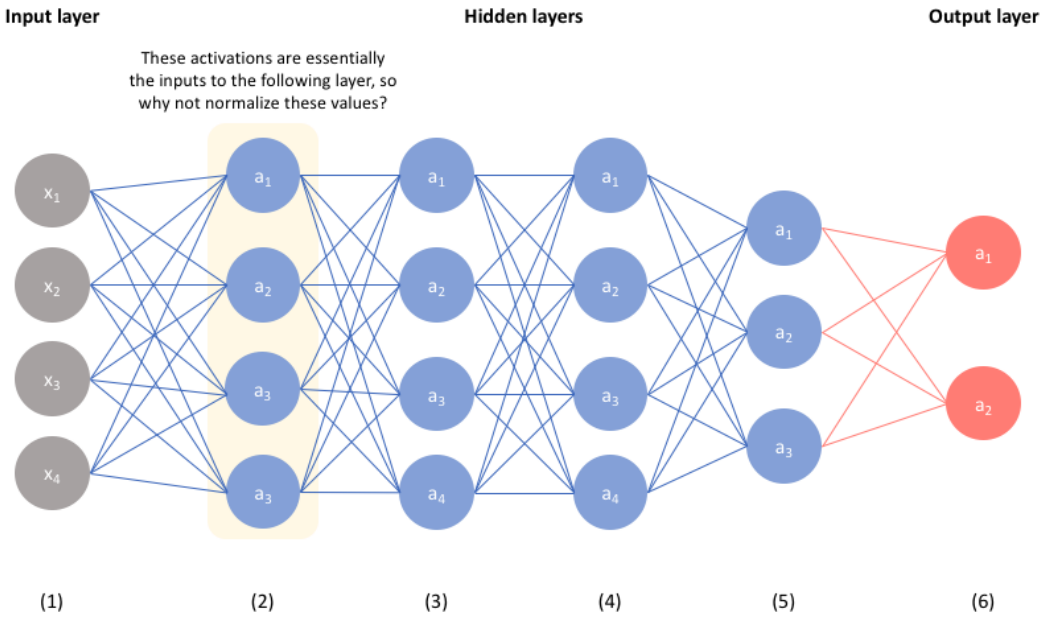


FIGURE 2.18: Example Neural Network(b) (Jordan, 2019)

helpful for the hidden layers of our network, since the distribution of unnormalized activations from previous layers will change as the network evolves and learns more optimal parameters. Thus, by normalizing each layer, we're introducing a level of orthogonality between layers - which generally makes for an easier learning process (Jordan, 2019)."

In machine learning, orthogonalization refers to tuning hyperparameters to achieve improved model results. Hyperparameters are model values that are set before the training process, such as learning rate, as opposed to other model parameters which are learned during the training process. Choice of hyperparameters can significantly affect the the speed and quality of model training and testing, and resulting model performance (Claesen and De Moor, 2015). In this case, hyperparameters are tuned within the context of a chain of assumptions: the model fits the training, validation, and the test set well on the cost function, and then generalizes to real world data, with consistent performance across these datasets. Within each step there are a distinct set of tuning functions one can use to improve the fit on the cost function: bigger

network (architecture), more advanced normalization/optimization method (such as Adam method for stochastic optimization) in the training set, regularization, or a bigger training set to improve model fit on the validation set, bigger validation set to better fit test set (if training and validation set fit the cost function well, but do not fit the test set well, this probably means the validation dataset is overfit), change the validation set or the cost function if previous steps all fit well but do not perform consistently on real world data. In contrast, tuning methods such as early stopping (method for regularization that involves ending model training early) make orthogonalization difficult because it simultaneously affects the training set (i.e. stopping the training early will reduce accuracy) as well as validation set performance (Ng, 2020). Normalization is a precise model tuning method that can be utilized as a pre-processing step on the input layer or within each hidden layer of the network on the training dataset through a process known as batch normalization.

2.3.3 Effects of Batch Normalization

Batch normalization is an extension of traditional input layer normalization that is applied to more parameters of a neural network (hidden layers). The previous section established that normalizing the inputs to the network help the network learn the parameters in the first layer more accurately and efficiently. Since the second and further layers of the network accept the activations from the previous layer, it is assumed that normalizing these values will also help the network learn more effectively (Ioffe and Szegedy, 2015). During training of a neural network, as the weights of a previous layer change, the distribution of the layer inputs to the next hidden layer will also change. This is known as internal covariance shift and results in slow training due to the need for smaller learning rates. Batch normalization helps eliminate covariance shift by normalizing the inputs to each hidden layer using the

mean and variance of the linear combinations of a batch of observations from the previous layer (Ioffe and Szegedy, 2015). It has been found that batch normalization allows for higher learning rates (faster training) and acts as a regularizer (avoiding model overfit), sometimes eliminating the need for older regularization techniques such as dropout. In fact, batch normalization does not drastically increase architecture complexity but it reaches state of the art results in a fraction of the training steps (Ioffe and Szegedy, 2015). However, recently, additional research has questioned why batch normalization works well for training neural networks, with previous research mostly focused on implementation and results only. Santurkar et al. (2018) argue that the success of batch normalization is not, in fact, due to the reduction in internal covariance shift but rather due to its impact on the loss function optimization landscape, resulting in smoother optimization topology. It was discussed in the previous section how a smoother topology allows for a more efficient gradient descent optimization towards the loss function minimum. The authors demonstrated empirically that there is no apparent connection between performance of batch normalization and reduction of internal covariance shift. The authors argue that their results should encourage a more "systematic investigation of the algorithmic toolkit of deep learning and the underpinnings of its effectiveness (Santurkar et al., 2018)."

Additionally, the introduction of batch normalization indicated that the use of dropout as a regularization may no longer be necessary. However, some research has further investigated the relationship between batch normalization and dropout. Dropout can be applied at different structural levels including neuron, channel, path, and layer level. Cai et al. introduced a framework in 2019 for analyzing these four dropout methods and the poor performance in convolutional neural networks (CNN), especially when used in the architecture after batch normalization. They found that the poor performance is due to incorrect placement of dropout in the network as well as using dropout

methods not well-suited for particular CNN tasks (Cai et al., 2019). Li et al. summarize additional guidelines for improved use of dropout and batch normalization. Interestingly, in the statistical experiments described in this paper to quantify the effect of dropout and batch normalization, the authors use one sentence to describe the image pre-processing, i.e. normalizing the data by using the channel means and standard deviations, but provide no additional context behind those choices (Li, Chen, Hu, and Yang, 2019). A 2018 paper by He et. al. (2019) was the only one found that used empirical study to quantify the effects of various CNN model tuning choices. They point out that many of these model refinements, including changes in loss functions, data pre-processing, and optimization methods have improved model accuracy but are rarely mentioned in detail outside of implementation details or source code. This study evaluated a dozen methods, including batch training, learning rate scaling, zero weight decay, and others, on multiple architectures and datasets, and found that several refinements lead to significant model improvement. While the studies mentioned above evaluate some internal model architecture normalization and optimization refinements, no studies were found that focus on statistical evaluation of data pre-processing in neural networks.

2.4 Effects of Model Frameworks and Normalization in Application

In relation to the effects of model development choices on downstream analysis, it is perhaps most interesting to consider the generalization of frameworks towards specific model applications.

2.4.1 NCAA Basketball Event Prediction

The NCAA Division I Men's Basketball has a 68 game single elimination tournament which is played annually, colloquially known as 'March Madness'. The popularity of the tournament has increased in recent years among basketball enthusiasts and data scientists (not necessarily mutually exclusive) due to the introduction of the popular March Madness Kaggle competition. In 2019, the longest streak of a perfect bracket to date was achieved when Gregg Nigl correctly predicted the first 49 games of the tournament. It should be noted that in the NCAA.com bracket tournament, points are awarded on a sliding scale with each round progressively awarding more points per game. This means that someone who correctly predicts the tournament winner, if no one else picks that same team, will automatically win the tournament even if the rest of their bracket performed poorly. In the NCAA.com game, players are simply rewarded for correct predictions. However, in the Kaggle competition, models are measured using the log loss function, with the aim to minimize log loss between predicted win probabilities and actual game outcomes. This loss function has high penalty for models that are both confident and wrong ("Applying Machine Learning To March Madness", n.d.). The choice of how one wants to compete in the March Madness competition (i.e. NCAA.com office pool versus data science community Kaggle competition) determines the evaluation metric and the resulting best-performing model development strategies. Paul Kvam and Joel Sokol at Georgia Tech developed what may be the current gold standard NCAA ranking model, which currently outperforms all other rankings and Las Vegas betting lines using only basic input data (Kvam and Sokol, 2006). The model, developed in 2005, uses a combined logistic regression/Markov chain (LRMC) to predict tournament games by ranking teams and estimating win probabilities, with logistic regression used to populate transition probabilities of the Markov chain. The LRMC

model was tested both as a standalone predictor as well as the source of a probability estimate for an existing dynamic programming model developed by Kaplan and Garstka (2001). Results showed that LRMC was better at predicting tournament winners than any other rankings and is significantly better at game-by-game predictions for all compared rankings except for the game-by-game Las Vegas odds. However, the Las Vegas odds use additional information that is not included in the LRMC or other rankings. In addition, in 4 of 5 test seasons, the LRMC model had the Final Four teams ranked collectively higher than any of the other rankings systems. The logistic regression described earlier also led to contradiction to conventional wisdom that good teams are more likely to win close games. Instead, it was found that teams that won a close home game were equally as likely to win the road game as teams that lost a close home game (between 33 to 36% road win rates). Overall, the LRMC model using only basic inputs predicts individual game outcomes better than standard rankings systems, is better at predicting Final Four teams, and is better than the selection committee's seedings. Instead of treating the outcome of games as binary win/lose events, LRMC estimates the probability that the winning team is better than the losing team based on game location and the margin of victory, so it is better at accurately predicting the outcome of close games. Brown et al. (2012) improved on the original LRMC model by finding that LRMC can be improved using an empirical Bayes model instead of the Logistic Regression. LRMC and over 100 other rankings were compared on a game-by-game basis; the Bayesian LRMC and classic LRMC scored better than all other systems. The Kvam and Sokol framework for NCAA tournament prediction provides a baseline for state-of-art tournament prediction models.

Considering the complexity of tournament prediction, concerns have been raised as to whether improved model development frameworks can be provided for these models. According to Yuan et al. (2015), modeling the wins as well

as the losses may attract endless statistical problems, and describe difficulties with forecasting games, including lack of historical data to train models, overfitting of models using post-tournament historical data, and high error on predictions due to highly variable team performance based on potentially unobservable factors. The authors tested over 30 models and found the most successful models incorporated regularization and did not suffer from data "contamination", i.e. archival data for a given season which incorporated the results of the final tournament for that year. Many publicly available datasets contain contaminated data and their use can result in overfit models. Popular aggregate statistics, including Pomeroy Ratings and Moore Power Rankings, are made of post-tournament rankings and are, therefore, a source of potential contamination. The authors of this paper made use of pre-tournament iterations of these datasets to avoid contamination, as well as study the magnitude and effects of contamination. Features were standardized by season before training the models, with the difference in competing teams' metrics used as model predictors, although no other normalization methods or their effects were considered in this research. They also found, through literature review and their own experimentation with a large set of features and predictive models, that parsimonious feature sets and simple algorithms tend to outperform complicated models with many features, as seen in Figure 2.19.

Most of the ensembles did not outperform the individual models, possibly due to overfitting or uncontrolled data contamination. The best performing models in this paper did not outperform a baseline 0.5 prediction. It was anecdotally suggested that predicting tournament games using regular season games would result in poor performance since these are two very different types of predictions. In addition, it is possible that since the available NCAA tournament data is relatively limited, it is not well suited to some more advanced algorithms such as random forests and neural networks. These limitations, in addition to the common forecasting difficulties the authors

Model	Log loss
A	0.67
B	0.61
C	0.71
D	0.98
E	0.84
Stacked (A, B, C, D, E)	0.77
Stacked (A, B, C)	0.75
Stacked (B, D)	0.63
Stacked (B, E)	0.86
All 0.5 benchmark	0.69
0.5+0.03* (Seed difference) benchmark	0.60
Mean Kaggle score	0.58

FIGURE 2.19: 2014 NCAA tournament log losses for logistic regression models (A, B, C), stochastic gradient boosted tree model (D), neural network (E), various stacked algorithms, and two naive benchmarks. The average Kaggle log loss score was 0.58. (Yuan et al., 2015)

previously described, could be addressed at various stages of the proposed unified model development framework.

Some researchers have proposed model frameworks for NCAA tournament prediction that have successfully implemented more complex models. Hao et. al. (2018) presented a framework for predicting bracket-based competitions using Recurrent Neural Networks (RNN) and combing with other models using combinatorial fusion, and describe three preprocess steps for feature selection including dataset transformation, merging highly correlated features, and selection and ordering of relevant features (see Figure 2.20). The differences in selected features between two teams are used in a recurrent neural network (RNN) model to predict winning probabilities of each game. This model was then compared to four classic models, with log loss results shown in Figure 2.21. Combining RNN with the 4 classic models using combinatorial fusion increased prediction accuracy from 70 to 75% but this provided no

performance improvement compared to the model we used in the 2019 tournament, which relied on the readily available Kaggle.com data and standard gradient boosting regression available in the Python scikit-learn package. While this study does suggest an important modeling framework to use for NCAA bracket prediction, it does so without truly quantifying some of the results of the use of such a framework. In fact, even though RNN has the best performance in the first phase, it only makes it into 2 of the 3 best combined models in the second phase, suggesting that combination of diverse models is more important than combination of best models but further quantification of this effect is required to confirm this suggestion.

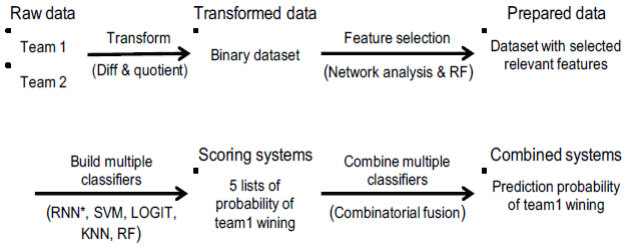


FIGURE 2.20: Game prediction framework as described in Hao et. al. (2018)

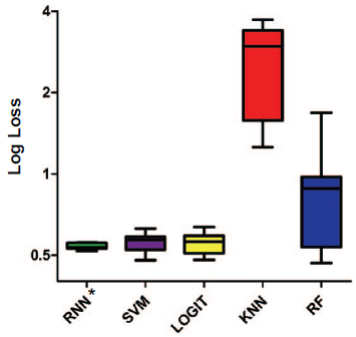


FIGURE 2.21: Log loss of five models in ten-fold cross validation (Hao, Kristal, and Hsu, 2018)

2.4.2 Normalization in Genetics

RNA-seq is a widely used method for studying the behavior of genes under different biological conditions. It is necessary to normalize data in an RNA-seq study to adjust data for factors that can prevent direct comparison between gene expression measures. Evans, Hardin, and Stoebel (Evans et al., 2017) presented a study on a sample of RNA-Seq normalization methods based on significant assumptions. RNA-seq normalization methods have been developed that address various experimental and gene expression assumptions. In this context, the raw data is adjusted to account for relevant factors said to prevent or impede direct comparison of the expression measures. It is worth noting that differences in gene expression identified in the course of normalization that may not be biologic in nature carry with them a significant impact on the downstream analysis, as seen in the inflated false positives across the differential expression analysis. For example, experimental variability such as variability in the total number of molecules sequenced can lead to different total read counts in different samples, i.e. differences in sequencing depth. When one sample has more reads than another, non-differentially expressed genes will tend to have higher read counts in that sample, requiring a correction. Normalization is required so that differences in normalized read count represent differences in true expression. In this research 'expression' and 'differential expression (DE)' refer to the absolute quantities of mRNA/cell, therefore the relationship between the normalized read counts must be known and correct. A gene is said to be differentially expressed (DE) if it produces different levels of mRNA/cell under different biological conditions. Previous studies (Bullard, Purdom, Hansen, and Dudoit, 2010) found that normalization procedure in DE had a larger impact on results than the choice of test statistic used in hypothesis tests.

One group of normalization methods used in RNA-seq analysis is normalization

by testing or distribution. Assumptions noted in these methods are that non-DE genes and the DE genes almost behave the same way, which implies that the technical effects are the same in the two cases. Another assumption is that of balanced expression, where normalization is said to tolerate significant differences in either up or down-regulated genes associated with higher proportions of DE. In fact, previous research by Robinson et. al. (Robinson and Oshlack, 2010) illustrated the importance of this last assumption. When first introduced, RNA-seq methods relied on normalizing by library size. However, this method is too limited for many biological applications, especially where a large number of genes are highly expressed in one experimental condition, resulting in differential expression analysis to be skewed towards one condition. The method presented by Robinson and Oshlack, 2010 "uses raw data to estimate appropriate scaling factors that can be used in downstream statistical analysis, i.e. a data-driven approach. A successful normalization method ensures that a gene with the same expression level in two samples will not be detected as DE." The authors used an empirical strategy to compare the expression levels of genes between samples under the assumption that the majority of them are not DE. This study illustrated the importance of normalization for RNA-seq data (even though RNA-seq was said to not require normalization to the same degree as older microarray technology), in "situations where the underlying distribution of expressed transcripts between samples is markedly different." The final group of normalization assumptions addressed by Evans et al. (2017) include normalization by controls, in which controls are defined when there is a violation of the assumptions in the previously mentioned methods. Assumptions required for this method include the existence of controls (for negative controls they are non-DE under the experimental conditions) and that the controls behave like non-control genes (technical effects are the same for controls and non-control genes so that controls can be used for normalization).

As discussed above, Evans et al. (2017) postulated that normalization choices based off of correct assumptions allow normalization to translate raw read counts into meaningful measure of expression (correct amount of fold-change relative between samples and conditions). The correct normalization method to use depends on valid experimental assumptions and incorrect normalization can lead to downstream analysis errors, including inflated false positives and untrustworthy results. While no normalization method is perfect, understanding the assumptions can lead towards the most suitable method for the given experimental conditions. Zyprych-Walczak et al. (2015) established that the final choice of the normalization approach would directly affect or influence the outcome of the DE analysis and that sensitivity is likely to vary between the test statistics but become more pronounced across the normalization procedures. In an effort to prove that normalization carries a significant impact on DE analysis, the researchers conducted an investigation that involved five normalization methods that are commonly applicable to RNA-seq data, and compared results on the analysis of three real RNA-seq data sets, two of which come from publicly available resources (Asmann et al., 2012, Cheung et al., 2010). The datasets vary with sample sizes, gene numbers, and gene expression levels. The authors evaluated normalization methods by first applying the bias and variance criterion proposed by Argyropoulos et al. (2006), adjusted to be suitable for RNA-seq data, with the bias reducing to the root mean square error (RMSE). The ratios of the mean bias and variance values for each method are computed for all control genes, and the preferred method is the one associated with smallest bias and variance (Figure 2.22). Classification errors were based on five classifiers: naïve Bayes, neural network, k-nearest neighbor, support vector machines, and random forest, utilizing leave one out cross validation (LOOCV). Their findings suggest that the choice of the normalization process can impact expression results. A poorly selected normalization method, or none at all, can lead to erroneous DE analysis.

Therefore, it is important to note that more effort needs to be put in place when conducting the pre-processing stage of analysis. The authors suggest a universal work flow for the selection of the optimal normalization procedure for any dataset including calculation of bias and variance for the control genes, sensitivity and specificity of the methods, and classification errors as well as diagnostic plots.

Concentration	20%	25%	30%	35%	40%	45%	50%	GMN	SBN
30 pg/μl	0.073	0.073	0.075	0.075	0.075	0.075	0.076	0.257	0.181
150 pg/μl	0.080	0.080	0.079	0.079	0.079	0.078	0.079	0.213	0.232
300 pg/μl	0.058	0.057	0.056	0.055	0.055	0.055	0.054	0.034	0.156
1500 pg/μl	0.060	0.060	0.058	0.057	0.057	0.056	0.055	0.115	0.161
3000 pg/μl	0.085	0.086	0.087	0.088	0.088	0.088	0.089	0.254	0.266
Average	0.071	0.071	0.071	0.071	0.071	0.071	0.071	0.175	0.199

RMSE is the square root of the average deviation and hence estimation bias of the true expression ratio, and spiked-in based normalization ranked second. SBN, spiked-in based normalization; GMN, global median normalization. A 20-50% lowess normalization with different choices of local neighborhood were used with the smoother.

FIGURE 2.22: Root mean square error (RMSE) performance of normalization algorithms. (Argyropoulos et al., 2006)

Research has also suggested the importance of normalization choices in microRNA (miRNA) analysis. Profiling the miRNA levels in a given cell is emerging as a dominant and widely used approach. However, there is no agreed upon consensus as to the best normalization to use and the relative performances of varying methodologies. Rai et. al. (2012) evaluated data quality, data normalization, and statistical hypothesis procedures used in miRNA samples of repeated subjects over time, suggesting that the intra-subject correlation created due to repeated sampling of patients and other key features of experimental design should to be incorporated into the analysis. In this study, the researchers were interested in an "exploratory analysis of changes in cardiac expression of miRNAs in patients with end-stage heart failure (HF) undergoing placement of a left ventricular assist device (LVAD) and subsequent heart transplantation." In this case, the measure of expression level is found using the Ct values, where "Ct value represents the cycle number at which the fluorescent signal of the reported dye crosses a threshold value." There are two unique challenges with the studied data: 1) in order

to maximize the number of unique miRNAs included in the experiment, technical replicates were not included so normal data quality techniques could not be considered. 2) repeated sampling of miRNAs from the same subjects over time present a normalization challenge as "typical normalization techniques are not designed to preserve naturally occurring correlation structure." They considered the delta-CT method, mean normalization, quantile normalization, and rank invariant normalization and compared the various normalization techniques, by calculating coefficient of variation (CV) for each miRNA over all the plates. Delta-Ct and mean normalization shift the mean expression value preserving the correlation structure. However, quantile normalization changes the distribution of the Ct values creating a new correlation structure while also reducing variance, shown in the improved results based on the coefficient of variation compared to the other methods. The authors question whether quantile normalization's effect on the correlation structure should be a concern specifically asking: what effect does it have on the generalized estimating equation (GEE) model and resulting significantly expressed genes? How does this effect compare to shift of center, invariant normalization procedures that do not reduce the variation in the Ct values? If the effect matters, should we just do the analysis on raw, unnormalized data? A more recent study by Schwarzenbach et al.(2015) also discussed the need for an optimal normalization strategy for miRNA analysis, pointing out that there is little to no consistency among normalization strategies for selecting endogenous and/or exogenous control reference genes. This has led to issues with "ambiguous data interpretation, misleading conclusions, and erroneous biological predicted affects" that leads to a lack of comparability and reproducibility between studies.

Chapter 3

Loss Function Decomposition and Model Characteristics

Before providing details as to the methods for building a quantitatively motivated pre-processing framework for predictive models, it is first necessary to explore the mathematical motivation behind such a framework. Here, we will explore the mathematical decomposition of several loss functions used to measure the effectiveness of predictive models (model error), as well as the characteristics of several models under consideration in the framework. Decomposing the loss functions into the pieces measuring bias, variance, and noise allows specific measurement of the bias-variance tradeoff when selecting appropriate pre-processing techniques and models for various statistical distributions and predictive modeling problem spaces. By measuring the specifics of these decompositions, we gain improved understanding and quantification of model and data elements affecting potential predictive issues including overfitting (high variance), underfitting (high bias), and, subsequently, model capacity. An important goal in model development is to control overall model error by minimizing bias and variance. In this context, we consider generalizations of Mean Square Error (MSE) and 0-1 Loss in terms of prediction, highlighting the relationships between machine learning bias and statistical bias and variance.

3.1 Relationship between Machine Learning and Statistical Measures of Error

According to Dietterich and Kong (1995) machine learning bias is a general term used to choose one model hypothesis over another. In this case, machine learning bias can be divided into two types: absolute bias and relative bias. Machine learning algorithms that consider absolute bias are those where "certain hypotheses are entirely eliminated from the hypothesis space", as in linear discriminant models, and algorithms that consider relative bias are those where "certain hypotheses are preferred over others", as in a decision tree algorithm such as CART where small trees are considered before larger ones (Dietterich and Kong, 1995). In this way, machine learning considers some amount of bias necessary in order to generalize algorithms for prediction. Without considering some absolute or relative machine learning bias, then we must consider all possible functions as hypotheses, and these functions then predict all possible outcomes equally, providing no useful information for generalization or prediction (Dietterich and Kong, 1995). For machine learning, the expectation of systematic error, bias, is due to the choice of model. By extension, variance in this case is understood to be the random error around the model approximations that are due to randomness in the training samples.

Statistical bias is a more specific term used to measure the expected error an algorithm or function will make when trained on samples of size m from the same probability distribution D. Traditionally, bias and variance are measured in terms of functions of estimators. That is, statistical bias and variance measure properties of functions, g, that estimate some characteristic of a population from a sample, S, of data drawn from this population. The estimator in this case is then:

$$\widehat{\Theta}_S = g(S), S = (x_1, ..., x_m), \tag{3.1}$$

where x_i is a random variable drawn from some probability distribution D where $x_i \sim D$. Statistical bias is then a property of how good this estimator, $\widehat{\Theta}_S$, is in estimating the real value of Θ. This is measured as the difference between the expected value of an estimator drawn from a sample, S, of size m, from distribution D and its true value in the population:

$$Bias(\widehat{\Theta}_S, \Theta) = \mathbb{E}_{S \sim D^m}[\widehat{\Theta}_S - \Theta]. \tag{3.2}$$

If the bias is less than zero we say the estimator is negatively biased, if the bias if greater than zero then the estimator is positively biased, and if the bias is zero the estimator is unbiased. Subsequently, the estimator variance is then a property of the consistency of this estimator among samples:

$$Var(\widehat{\Theta}_S) = Var_{S \sim D^m}[\widehat{\Theta}_S]. \tag{3.3}$$

More specifically, statistical variance is the difference between the expected value of the squared estimator and the squared expectation of the estimator:

$$Var(\widehat{\Theta}) = \mathbb{E}[\widehat{\Theta}^2] - (\mathbb{E}[\widehat{\Theta}])^2 \tag{3.4}$$

which is equivalently, since expectations are linear functions and can be distributed:

$$Var(\widehat{\Theta}) = \mathbb{E}[(\mathbb{E}[\widehat{\Theta}] - \widehat{\Theta})^2]. \tag{3.5}$$

Mean squared error (MSE) of the estimator is then a combination of these estimator properties as:

$$MSE = \mathbb{E}[(\widehat{\Theta}_S - \Theta)^2] = Bias^2(\widehat{\Theta}_S, \Theta) + Var(\widehat{\Theta}_S) \tag{3.6}$$

where expectations are taken with respect to the sample, S, and parameter Θ.

Proof.

$$\begin{aligned} MSE &= \mathbb{E}[(\widehat{\Theta}_S - \Theta)^2] \\ &= \mathbb{E}[\widehat{\Theta}_S^2 - 2\widehat{\Theta}_S\Theta + \Theta^2] \\ &= \mathbb{E}[\widehat{\Theta}_S^2] - 2\mathbb{E}[\widehat{\Theta}_S]\Theta + \Theta^2 \\ &= \mathbb{E}[\widehat{\Theta}_S^2] - \mathbb{E}[\widehat{\Theta}_S]^2 + \mathbb{E}[\widehat{\Theta}_S]^2 - 2\mathbb{E}[\widehat{\Theta}_S]\Theta + \Theta^2 \\ &= \mathbb{E}[\widehat{\Theta}_S^2] - \mathbb{E}[\widehat{\Theta}_S]^2 + (\mathbb{E}[\widehat{\Theta}_S] - \Theta)^2 \\ &= Var(\widehat{\Theta}_S) + Bias^2(\widehat{\Theta}_S, \Theta) \end{aligned} \tag{3.7}$$

□

Where in statistics we use bias and variance to measure properties of estimators that estimate a characteristic of a population, in machine learning we're interested in predicting values using samples of data. In supervised machine learning, a model A is used to learn some target function f, where f maps data $\mathbf{X}$ to some real number $\mathbb{R}$ prediction. As with the parameter estimation above, we're interested in samples of size m coming from probability distributions, D. In this case, D is the probability distribution over the input space $\mathbf{X}$ such that a random sample from the input space, $x \in X$, is drawn with probability $D(x)$. Then the sample, S, of size m, drawn from D is:

$$S = (x, f(x) + \epsilon) | x \in X \tag{3.8}$$

where each random example, x, in the sample is labeled with the value of the learned function f, plus some noise ϵ. So, for each sample, the model A will output a hypothesis of the learned function f. For a given example within a sample, x_0, the predicted value of the function is $\widehat{f}(x_0)$.

Since our goal in machine learning is to find an approximation of $f(x)$ since we cannot observe this value directly, we instead draw repeated samples,

$S_1, ..., S_l$, of size m, to arrive at this approximation by building an averaged hypothesis with model A over $\widehat{f}_{S_1}, \widehat{f}_{S_2}, ...\widehat{f}_{S_l}$, where $\widehat{f}_{S_l}$ is the model A learned function f hypothesis in sample i. This averaged prediction is then:

$$\overline{\widehat{f}}(x) = \lim_{l \to \text{inf}} \frac{1}{l} \sum_{i=1}^{l} \widehat{f}_{S_i}(x) \tag{3.9}$$

Since $\overline{\widehat{f}}$ is the expected predicted value of $f(x)$ over different samples of size m, we can instead call this expected value as the average, $\mu(x)$, over different possibilities of the training set, τ, and write the average prediction, over all training sets as:

$$\mu(x) = \mathbb{E}_\tau[\widehat{f}_\tau(x)]. \tag{3.10}$$

The bias of model A for sample size m at example point x is then the error in this average prediction:

$$Bias(A, m, x) = \overline{\widehat{f}}(x) - f(x) \tag{3.11}$$

which can be written alternatively as:

$$\begin{aligned} Bias(x) &= \mathbb{E}_\tau[\widehat{f}_\tau(x) - f(x)] \\ &= \mu(x) - f(x). \end{aligned} \tag{3.12}$$

Bias is capturing systematic error which comes from the choice of model A learning $f(x)$. Since we're interested in the approximation of $\widehat{f}(x)$ over many training sets, τ, we know this model won't be an exact match to the true population data distribution.

Variance in this case is similar to that from statistical parameter estimation variance, and it measures the average distance between the real function and the predicted function. Variance is random error that results from "variation in the training sample, random noise (ϵ) in the training data, or from random

behavior in the learning algorithm itself, such as the random initial weights often used in backpropagation (Dietterich and Kong, 1995)." Specifically, variance of algorithm A is the expected value of the squared difference between the predicted function $\widehat{f}_S$ and the average predicted function $\overline{\widehat{f}}$ over the sample space S:

$$Var(A, m, x) = \mathbb{E}[(\widehat{f}_S - \overline{\widehat{f}}(x))^2]. \tag{3.13}$$

Since the expectation is taken with respect to all training samples S of size m then we can also rewrite the variance as we did with the bias as:

$$Var(x) = \mathbb{E}_\tau[(\widehat{f}_\tau(x) - \mu(x))^2]. \tag{3.14}$$

Since expectations are linear functions we can distribute the expectation and write:

$$Var(x) = \mathbb{E}_\tau[\widehat{f}_\tau(x)^2] - \mu(x)^2. \tag{3.15}$$

To illustrate the relationship of statistical bias and variance to machine learning, let's assume there's an unknown function we'd like to approximate. We draw several different training sets from an unknown distribution and define these sets in terms of the unknown target function plus noise or irreducible error. Figure 3.1 includes several linear regression models fit to different training sets. The predicted linear regression functions do not fit well to the true function except at two points. This illustrates a high bias model since the difference between the predicted value and the true value, on average (the expectation over the training sets, not the average of the examples within each training set), is very large.

Now suppose instead we fit unpruned decision tree models to several training sets. In Figure 3.2 we can see these predicted models fit the training data very closely. In fact, the expectation over the training sets would result in the the average prediction fitting the true function perfectly, since noise is

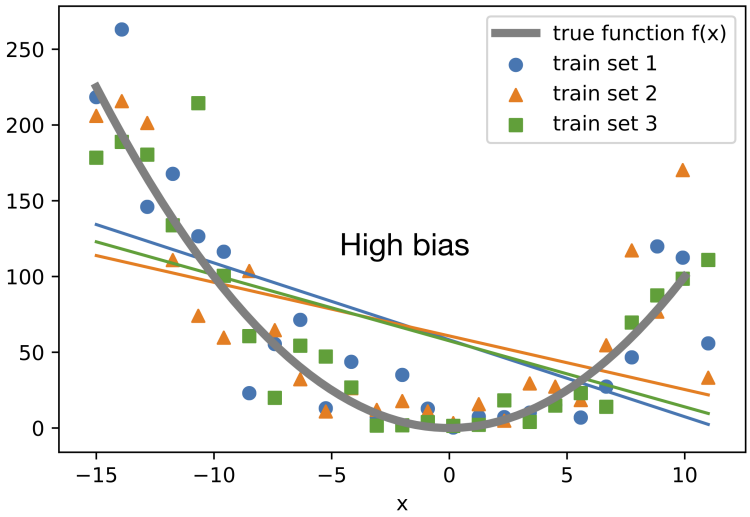


FIGURE 3.1: Illustration of High Bias (Raschka, n.d.)

assumed to be unbiased with an expected value of 0. However, the variance in this case is very high since, on average, the predictions differ greatly from expected value of the true function.

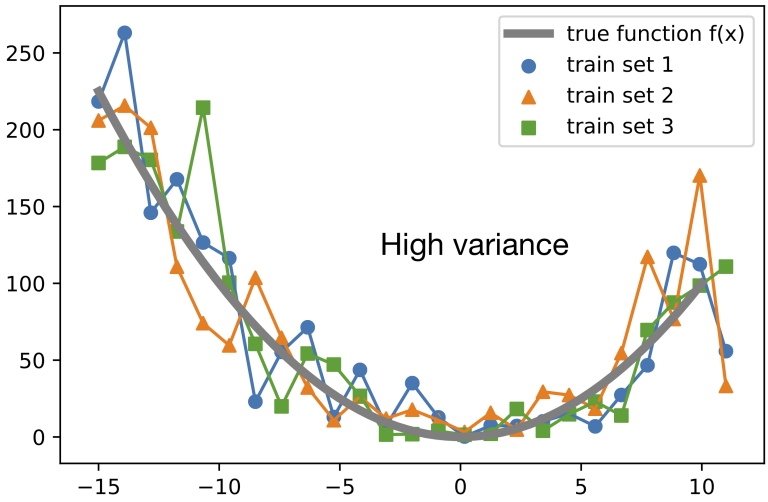


FIGURE 3.2: Illustration of High Variance (Raschka, n.d.)

The figures illustrate what is commonly referred to as the "bias-variance tradeoff," indicating that modifying some aspect of the learning algorithm often has opposite effects on the bias and the variance; as one increases the complexity of the algorithm (increasing the degrees of freedom - number of parameters that need to be estimated), the bias decreases but the variance increases. The goal is to optimize the learning algorithm in such a way

that decreases expected loss by optimizing this trade-off between bias and variance.

3.2 Generalization of MSE Decomposition for Prediction

Consider a regression problem with an observed outcome Y and set of predictors X. We can define the relationship between X and Y as:

$$Y = f(X) + \epsilon \tag{3.16}$$

where f is the unknown model that maps the predictor X to the true outcome $f(X)$, and ϵ is an error term for observation noise. Since f and ϵ are unknown, we use sets of training data, τ, to estimate the function and predict the outcome as $\widehat{f}_\tau(X)$. We can then use the squared-error loss to predict the error based on a test set, with the goal to minimize the following:

$$MSE = \mathbb{E}_{X,Y} \sim D, \tau \sim D^m, \epsilon \sim E[(Y - \widehat{f}_\tau(X))^2] \tag{3.17}$$

where (X,Y) is the unobserved data following distribution D, τ is the data sets of size m used to train the predictor which also follow distribution D, and ϵ is the observation noise which follows some distribution E. It is important to note that the unobserved (testing) data comes from the same distribution as the training data τ. The squared-error can be rewritten as:

$$\mathbb{E}[(Y - \widehat{f}(X))^2] = \mathbb{E}[f(X) + \epsilon - \widehat{f}_\tau(X))^2] \tag{3.18}$$

In this case, squared error acts as a risk function rather than a loss function. Specifically, *Loss(L)* is a measure of how well a model using $\widehat{f}(X)$ approximates the true value of $f(X)$ using the training data. However, our goal is to fit models that generalize well to unseen data, and not just to the training data

(which can lead to overfitting). In this case, we're interested in the average measure of loss (expected loss) across the whole data distribution, which we approximate using repeated samples of training data and predicting on a test dataset. *Loss* defined in this way using the above equation is specifically known as a *risk function*. Since we are approximating the true data distribution using repeated samples of training data, minimizing the loss function and minimizing risk are approximately the same. For the rest of the discussion on bias-variance decomposition, *loss functions* will be used to describe the decompositions, except when we specifically refer to expected values using repeated samples of training data and applied to test data, in which case *risk function* terminology will be used. Before performing the bias-variance decomposition we must first understand several factors. First, the decomposition is performed in reference to the test set so X and ϵ are those values from the test set. Also, the observed value of Y is dependent on both X and ϵ, whereas the predicted outcome, $\widehat{f}_\tau(X)$ is dependent on the training set τ used for the estimation as well as X. We are also assuming during this process that ϵ has a mean of zero. Finally, we will use the following identities of variance, covariance, and expectations to complete the decomposition:

$$\begin{aligned} Var(X) &= \mathbb{E}[X^2] - \mathbb{E}^2[X] \\ \mathbb{E}[XY] &= \mathbb{E}[X]\mathbb{E}[Y] + Cov(X,Y) \\ Var(X+Y) &= Var(X) + Var(Y) + 2Cov(X,Y) \\ Var(X-Y) &= Var(X) + Var(Y) - 2Cov(X,Y) \\ Cov(X,Y) &= 0 \text{ if X and Y are independent} \end{aligned} \tag{3.19}$$

The squared-error can now be written as:

$$\mathbb{E}_\tau[\mathbb{E}_{X,\epsilon}[(Y - \widehat{f}_\tau(X))^2]] = \mathbb{E}_\tau[\mathbb{E}_{X,\epsilon}[(f(X) + \epsilon - \widehat{f}_\tau(X))^2]] \tag{3.20}$$

Since X and ϵ are independent we can expand the expression to be:

$$\mathbb{E}_\tau[\mathbb{E}_{X,\epsilon}[(Y - \widehat{f}_\tau(X))^2]] = \mathbb{E}_X[\mathbb{E}_\tau[\mathbb{E}_\epsilon[(f(X) + \epsilon - \widehat{f}_\tau(X))^2]] \tag{3.21}$$

The inner-most expectation, assuming ϵ has a mean of zero, is:

$$\begin{aligned}
\mathbb{E}_\epsilon[(Y - \widehat{f}_\tau(X))^2] &= \mathbb{E}_\epsilon[(f(X) + \epsilon - \widehat{f}_\tau(X))^2] \\
&= \mathbb{E}_\epsilon[(f(X) - \widehat{f}_\tau(X))^2 + \epsilon^2 + 2\epsilon(f(X) - \widehat{f}_\tau(X))] \\
&= (f(X) - \widehat{f}_\tau(X))^2 + \mathbb{E}_\epsilon[\epsilon^2] + 2(f(X) - \widehat{f}_\tau(X))\mathbb{E}_\epsilon[\epsilon] \\
&= (f(X) - \widehat{f}_\tau(X))^2 + (\mathbb{E}_\epsilon[\epsilon^2] - \mathbb{E}_\epsilon[\epsilon]^2) + E_\epsilon[\epsilon]^2 \\
&+ 2(f(X) - \widehat{f}_\tau(X))E_\epsilon[\epsilon] \\
&= (f(X) - \widehat{f}_\tau(X))^2 + Var_\epsilon[\epsilon] + 0 + 0 \\
&= (f(X) - \widehat{f}_\tau(X))^2 + Var_\epsilon[\epsilon]
\end{aligned} \tag{3.22}$$

The second inner-most expectation is:

$$\begin{aligned}
\mathbb{E}_\tau[\mathbb{E}_\epsilon[(Y - \widehat{f}_\tau(X))^2]] &= \mathbb{E}_\tau[(f(X) - \widehat{f}_\tau(X))^2 + Var_\epsilon[\epsilon]] \\
&= \mathbb{E}_\tau[(f(X) - \widehat{f}_\tau(X))^2] + Var_\epsilon[\epsilon] \\
&= \mathbb{E}_\tau[f(X)^2 + \widehat{f}_\tau(X)^2 - 2f(X)\widehat{f}_\tau(X)] + Var_\epsilon[\epsilon] \\
&= f(X)^2 + \mathbb{E}_\tau[\widehat{f}_\tau(X)^2] - 2f(X)\mathbb{E}_\tau[\widehat{f}_\tau(X)] + Var_\epsilon[\epsilon] \\
&= f(X)^2 + \mathbb{E}_\tau[\widehat{f}_\tau(X)^2] - 2f(X)\mathbb{E}_\tau[\widehat{f}_\tau(X)] + Var_\epsilon[\epsilon] \\
&+ \mathbb{E}_\tau[\widehat{f}_\tau(X)]^2 - \mathbb{E}_\tau[\widehat{f}_\tau(X)]^2 \\
&= (f(X)^2 + \mathbb{E}_\tau[\widehat{f}_\tau(X)]^2 - 2f(X)\mathbb{E}_\tau[\widehat{f}_\tau(X)]) + (\mathbb{E}_\tau[\widehat{f}_\tau(X)^2] \\
&- \mathbb{E}_\tau[\widehat{f}_\tau(X)]^2) + Var_\epsilon[\epsilon] \\
&= (f(X) - \mathbb{E}_\tau[\widehat{f}_\tau(X)])^2 + Var_\tau[\widehat{f}_\tau(X)] + Var_\epsilon[\epsilon]
\end{aligned} \tag{3.23}$$

Then finally we can solve the outer-most expectation and get:

$$\begin{aligned}\mathbb{E}_X[\mathbb{E}_\tau[\mathbb{E}_\epsilon[(Y-\widehat{f}_\tau(X))^2]]] &= \mathbb{E}_X[(f(X)-\mathbb{E}_\tau[\widehat{f}_\tau(X)])^2 + Var_\tau[\widehat{f}_\tau(X)] + Var_\epsilon[\epsilon]] \\ &= \mathbb{E}_X[(f(X)-\mathbb{E}_\tau[\widehat{f}_\tau(X)])^2] + \mathbb{E}_X[Var_\tau[\widehat{f}_\tau(X)]] + Var_\epsilon[\epsilon]\end{aligned} \tag{3.24}$$

In this way we can see that the squared-error risk function for prediction decomposes into $Var_\epsilon[\epsilon]$, the irreducible error/random observation noise, and the reducible error caused by the algorithm. The reducible error can be broken into $\mathbb{E}_X[(f(X)-\mathbb{E}_\tau[\widehat{f}_\tau(X)])^2]$, the average bias, and $\mathbb{E}_X[Var_\tau[\widehat{f}_\tau(X)]]$, the average variance. Bias for a prediction problem is the amount by which our estimated function $\hat{f}_\tau(X)$ differs from the true value $f(X)$ and is caused by choice of model, e.g. approximating a very complicated relationship with too simple of a model. Variance for a prediction problem, on the other hand, is the amount an estimate of the function $\hat{f}_\tau(X)$ differs from the average value of the function over all test sets. Once again, we can see that to minimize the test error we need to select a model with both low bias and variance. However, we often have to balance the trade-off of selecting a more complex model, resulting in low bias but higher variance (Zeng, 2018). For example, high capacity models such as neural networks have low bias since they approximate the real model function very well, but they have high variance since it is more difficult to generalize a model from training data to new test data. Lower capacity models such as regression have high bias but low variance.

3.2.1 Loss Function Decomposition for Classification

The decomposition for prediction provided in the previous section has been a generalized decomposition for regression models. For classification problems, a quadratic loss function is often inappropriate since the class labels are not numeric. From the computer science/machine learning perspective, many

instead use misclassification rate or, equivalently, 0-1 loss (Kohavi, Wolpert, et al., 1996). To extend this idea to classification tasks we must assume that f is now a function that maps input data X to a finite set of class labels $c1, c2, ...ck$ and that, once again, given a sample S of size m, model A will output prediction function $A(S) = \widehat{f_S}(X)$. For a classification problem, we then want to know the probability that $\hat{f}_S$ misclassifies a test point x as $\hat{p}_S(x)$ with:

$$\hat{p}_S(x) = \begin{cases} 1 & if \hat{f}_S(x) \neq f(x) \\ 0 & if \hat{f}_S(x) = f(x) \end{cases} \tag{3.25}$$

Since we don't know the true value of $f(x)$, we estimate the value by applying the model over a set (τ) of training samples $S_1, ...S_l$ each of size m with predicted functions $\hat{f}_{S_1}, \hat{f}_{S_2}, ..., \hat{f}_{S_l}$. Then the average probability of misclassification over the training set becomes:

$$\bar{\hat{p}}(A, m, x) = \lim_{l \to \text{inf}} \frac{1}{l} \sum_{i=1}^{l} \widehat{p}_{S_i}(x) \tag{3.26}$$

Generally, $\bar{\hat{p}}(A, m, x)$ is the probability that a predicted function from model A from a training set of size m will misclassify test point x. If a point x has $\bar{\hat{p}}(A, m, x) > 0.5$ then, on average, $\hat{f}_S$ will misclassify the test point. This is a systematic error and leads to our understanding of bias for a classification problem to be:

$$Bias(A, m, x) = \begin{cases} 0 & if \bar{\hat{p}}(A, m, x) \leq 0.5 \\ 1 & if \bar{\hat{p}}(A, m, x) > 0.5 \end{cases} \tag{3.27}$$

Variance of model A at point x is then the difference between the average probability of error and the bias:

$$Variance(A, m, x) = \begin{cases} \bar{\bar{p}}(x) & if \bar{\bar{p}}(x) \leq 0.5 \\ \bar{\bar{p}}(x) - 1 & if \bar{\bar{p}}(x) > 0.5 \end{cases} \tag{3.28}$$

3.2.2 0-1 Loss Decomposition for Classification

The previous section describes a generalized 0-1 loss function for classification in relation to the squared-error bias-variance decomposition as described by Dietterich and Kong (1995). However, it suffers from some shortcomings, including potentially negative variance, and it doesn't relate well to the MSE decomposition which is a strictly additive decomposition. An updated decomposition has been provided by Domingos (2000) and provides an improved bias-variance decomposition of the 0-1 loss that is directly related to the standard squared error loss decomposition. The refined definitions of bias and variance, applicable to any loss function, and a resulting decomposition for 0-1 loss, includes weighted factors on bias and variance. These factors resolve to an additive effect of variance in unbiased examples and a subtractive effect in biased ones. Table 3.1 below provides a summary of the relationship of relevant terms between MSE and 0-1 loss.

	Squared Loss	**0-1 Loss**
Single loss	$(y - \hat{y})^2$	$L(y, \hat{y})$
Expected loss	$\mathbb{E}[(y - \hat{y})^2]$	$\mathbb{E}[L(y, \hat{y})]$
Main prediction $\mathbb{E}[\hat{y}]$	mean (average)	mode
$Bias^2$	$(y - \mathbb{E}[\hat{y}])^2$	$L(y, \mathbb{E}[\hat{y}])$
Variance	$\mathbb{E}[(\mathbb{E}[\hat{y}] - \hat{y})^2]$	$\mathbb{E}[L(\hat{y}, \mathbb{E}[\hat{y}])]$

TABLE 3.1: Relationships between relevant terms of loss functions (Raschka, n.d.)

One of the main differences between squared error loss and 0-1 loss is that the main prediction for MSE is the expected average of all predictions over

the training sets, $\mathbb{E}_\tau[\hat{y}]$, and for 0-1 loss it is simply the mode, i.e. if the model predicts a label more than 50% of the time on average over all training sets, then the prediction will be 1, otherwise it will be 0. In this case, if the 0-1 loss is defined by a prediction $\hat{y}$ using the mode, then if the prediction does not agree with the true value of y the bias will be 1, and 0 otherwise.

$$Bias = \begin{cases} 1 & if y \neq \mathbb{E}[\hat{y}], \\ 0 & otherwise. \end{cases} \tag{3.29}$$

By extension, the variance is the probability that the predicted value $\hat{y}$ does not match the expected value of the prediction calculated over the training set τ:

$$Variance = P(\hat{y} \neq \mathbb{E}[\hat{y}]). \tag{3.30}$$

If we assume that ϵ has a mean of zero, and assume that loss = bias + variance, then we can show what happens to 0-1 loss if bias is 0:

$$Loss = 0 + Variance => Loss = P(\hat{y} \neq y) => Variance = P(\hat{y} \neq \mathbb{E}[\hat{y}]). \tag{3.31}$$

Considering that a model with 0 bias is likely to suffer from overfitting, it makes sense that 0-1 loss with no bias is completely defined by its variance. A less intuitive scenario is when bias is equal to 1, the average prediction on the test set is always wrong. We can rewrite the 0-1 loss as:

$$Loss = P(\hat{y} \neq y) = 1 - P(\hat{y} = y). \tag{3.32}$$

If we know bias is 1 then $y \neq \mathbb{E}[\hat{y}]$, and $y = \hat{y}$, so $\hat{y} \neq \mathbb{E}[\hat{y}]$ and:

$$Loss = P(\hat{y} \neq y) = 1 - P(\hat{y} = y) = 1 - P(\hat{y} \neq \mathbb{E}[\hat{y}]). \tag{3.33}$$

Therefore, when the bias is equal to 1, the loss is defined as $loss = bias - variance$ or $loss = 1 - variance$. This leads to the situation where, when the average prediction on the test set is always wrong ($bias = 1$), increasing the variance can actually decrease the loss. Both Dietterich and Kong (1995) and Domingos (2000) explain the intuition behind this by pointing out that in models with very high bias, increasing the variance can move the decision boundary, resulting in some correct prediction simply by chance.

3.3 Pre-Processing Effects on Loss Function Decomposition

At this point in our consideration of loss function decomposition it is valuable to consider the expected effects of certain pre-processing choices, specifically data normalization procedures, on the overall loss and the decomposed bias-variance. While certain choices may not effect the overall loss, it is still possible to find granular effects on the bias variance. This is important to understand particularly in situations where an algorithm choice has a known effect on bias and/or variance (e.g. a nearest neighbor model resulting in low bias but high variance) and a normalization procedure can provide quantifiable improvements to these expected effects (e.g. a normalization technique that has little or no effect on bias but improves variance). Since we are considering both the traditional statistical loss function of MSE as well as the common ML loss function of misclassification (0-1 loss), then we will consider the effects of normalization on both loss functions. In addition, we will consider the effects of data invariant as well as data variant normalization techniques. For example, techniques such as z-score standardization (transforms data to have a mean of zero and a standard deviation of 1) and feature scaling (rescaling data to have values between 0 and 1) change the spread and position of data points (all by consistent factors) but do not change the distribution shape of the data. While these simple normalization techniques should not affect

the overall loss, they can affect the more granular results of the bias-variance decomposition. However, for techniques such as quantile normalization and upper quartile normalization, both commonly used in genetic differential expression analysis, the measures of spread, position, and shape are all affected.

3.4 Quantifying Bias-Variance Tradeoff

We can measure the effects of various pre-processing methods on the bias-variance decomposition of the loss functions by directly simulating the two definitions of the decomposition under varying conditions. We can use information found from these simulations to propose decision points in a pre-processing framework for the predictive models of interest. Since "an important goal in algorithm design is to minimize statistical bias and variance and thereby minimize error (Dietterich and Kong, 1995)," we can use our findings to propose pre-processing and algorithm design choices that best minimize common design effects on bias and variance. For example, "any change that increases the representational power of an algorithm can reduce its statistical (and ML) bias. Any change that expands the set of available alternatives for an algorithm or makes them depend on a smaller fraction of the training data can increase the variance of the algorithm (Dietterich and Kong, 1995)." The result of such a study is to formulate a theory of bias and variance reduction and predict when either or both will succeed in practice.

3.5 Model Characteristics

In order to test the effects of various pre-processing methods on select data structures, the data structures and normalizations are considered under a selection of commonly used modeling techniques. The selected models discussed below represent a range of simple to complex, parametric and non-parametric,

global and local, stochastic methods. Each method has characteristics that may require normalization for optimal results or lead to unintended effects if incorrect normalization is used. The selected models and their effects are discussed below.

3.5.1 Generalized Linear Models

Historically, Generalized Linear Models (GLM) are an extension of simple linear regression models with continuous targets and continuous and/or categorical features. The form of such a model is expressed as

$$y_i \sim N(x_i^T \beta, \sigma^2), \tag{3.34}$$

where x_i is the data in feature i, and β are the coefficent parameters to be estimated as part of the linear function. In simple linear regression the assumption is that y is normally distributed, and the errors are normally distributed as $e_i \sim N(0, \sigma^2)$ and independent, the data is fixed, and there is constant variance σ^2. The GLM extends this simple linear model concept by assuming the target variable, y_i, follows a distribution within the exponential family (i.e. normal, binomial, poisson, etc.) with mean μ_i. The target then follows some linear or nonlinear function of $x_i^T \beta$, the linear combination of data and estimated coefficient parameters (Agresti, 2003). A summary of common GLMs is found in Table 3.2.

Model	Random	Link	Systematic
Linear Regression	Normal	Identity	Continuous
ANOVA	Normal	Identity	Categorical
ANCOVA	Normal	Identity	Mixed
Logistic Regression	Binomial	Logit	Mixed
Loglinear	Poisson	Log	Categorical
Poisson Regression	Poisson	Log	Mixed
Multinomial response	Multinomial	Generalized Logit	Mixed

TABLE 3.2: Summary of common Generalized Linear Models from Agresti

Generalized Linear Models are comprised of three main components: Random, Systematic, and Link Function. The random component refers to the distribution of the target variable (Y), e.g. normal distribution in linear regression, or binomial distribution in logistic regression. The systematic component specifies the explanatory features ($X_1, X_2, ...X_k$) and their linear combination. The Link Function specifies the link between the random distribution of the target variable and the systematic features. Assumptions of GLMs include:

- Data are independently distributed
- Errors are independent, but do not need to be normally distributed (i.e. Logistic Regression)
- Dependent variable does not need to be normally distributed (expect in linear regression) but are distributed within the exponential family
- Assumes a linear relationship between the link function transformed target and the explanatory features
- Uses Maximum Likelihood Estimation (MLE) to estimate the parameters, so it relies on large sample properties and regularity conditions (1st and 2nd derivatives must exist)

GLMs use Maximum Likelihood Estimation (MLE) to estimate the model parameters. In each of the distributions considered above (i.e. Linear, Logistic,

Poisson, etc.), the distribution depends on one or more unknown parameters, θ. The value of these parameters, θ, is estimated using observed data x. The function of θ that results from plugging in observed data x is known as the Likelihood Function:

$$L(\theta; x) = \prod_{i=1}^{n} f(X_i; \theta) \tag{3.35}$$

This function is the product of the values of the parameters, given each sample of data, and is denoted simply as $L(\theta)$. The log-likelihood is often used for computational convenience. The goal in GLMs is to maximize the likelihood of a parameter estimate given the observed data. The value of θ that maximizes this function is known as $\hat{\theta}$, the maximum-likelihood estimate (MLE). The maximum of the function is found by taking the derivatives with respect to the parameter(s) θ.

In this work, three generalized linear models are considered for three distinct target data types: linear regression, logistic regression, and Poisson regression.

Linear Regression

Linear regression is used for data with a continuous target which is a linear combination of the explanatory features, as in

$$Y_i = \beta_0 + \beta x_i + \epsilon_i \tag{3.36}$$

where index i represents each data point. This models the mean expected value of Y. The random component of linear regression, Y, has a normal distribution and normally distributed errors, $e_i \sim N(0, \sigma^2)$. The systematic component, the explanatory features X, can be continuous, categorical, or a combination of both, and is linear in the parameters $\beta_0 + \beta_i$. In multiple linear regression with multiple explanatory features, there is still a linear combination of the features in terms of their coefficient parameters β's but

the features themselves can have transformations, i.e. X^2 or $log(X)$. The link function is the identity link, $\eta = E(Y_i)$ since linear regression is modeling the mean response directly.

Logistic Regression

When there is a binary target (i.e. 0 and 1) binary logistic regression models the log odds of probability of "success" (target = 1). The random component, Y, has a binomial distribution, $Binomial(n, \pi)$, where π is the probability of success. The systematic component, X, can be continuous, categorical, or a combination of both, and is also linear in the parameters as in linear regression. However, in this case, the link function is the Logit link, $\eta = logit(\pi) = log(\frac{\pi}{1-\pi})$. Specifically, the logit link models the log odds of the mean response, π.

Poisson Regression

When the target of interest is an expected count (i.e. counts of disease, number of homes sold in a day, etc.), we extend the generalized linear model to use a log-linear or Poisson regression model. This models the expected count as a function of the explanatory predictors, $X = (X_1, X_2, ...X_k)$, where the predictors can be continuous, categorical, or a combination of both. When all the predictors are categorical this is known as a log-linear model. The random component of the Poisson model is the response Y with Poisson distribution, $y_i \sim Poisson(\mu_i)$ for $i = 1, ..., N$ where expected count of y_i is $E(Y) = \mu$. The systematic component is, as in the other GLM models, the linear combination of explanatory features X. Finally, the link function for the Poisson regression model is the natural log link, $log(\mu) = \beta_0 + \beta_1 x_1$.

Advantages of GLM

- Do not need to transform target variable to have normal distribution

- Models fit using MLE which provides statistically optimal properties of the estimators
- Model can be easily explained and parameters can be interpreted in the context of the prediction problem
- Easily implemented in most software

Disadvantages of GLM

- Still has to be a linear function of the parameters; the link function serves only to connect the nonlinear target distribution to a linear function
- Target responses must be independent

3.5.2 Decision Tree

A decision tree is a non-parametric classification technique that learns decision rules from features, using locally optimized, recursive partitioning. The algorithm assigns each sample in a dataset into a predicted class based on each samples' feature attributes. The algorithm uses information gain (3.37) to find the best features for classifying the data, where p and n are the proportion of 0 and 1 values of a binary outcomes for the i-th target class. Then, for each value defined for the decision values of the best feature (the feature and splitting value that best splits the predicted 0 and 1 outcomes), the algorithm repeats the process with additional, next-best predictive features. This process continues until the leaves of the tree are pure (samples at each node belong to the same class) or a pre-defined stopping criteria is reached (Owen, Ryza, Laserson, and Wills, 2015). In this way, decision tree is also a feature importance algorithm, where the data will be split on the most important, predictive features first.

$$G(A) = I(p,n) - \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i) \tag{3.37}$$

where

$$I(p,n) = -\frac{p}{p+n}log_2\frac{p}{p+n} - \frac{n}{p+n}log_2\frac{n}{p+n}$$

Advantages

- Since the decision tree algorithm is based on ordering and splitting the values within each feature, rather than a scale-dependent maximum likelihood optimization, scaling and normalizing features is not required
- Robust to missing data
- This model provides visual splits of the data and ordered feature importance that is easy to understand and interpret
- Implicit variable screening and selection – the top nodes of the tree are the most important variables in the dataset
- Non-parametric model does not assume linearity or any other distribution of the data. Model is built only based on observed data

Disadvantages

- Since this is a locally optimized, greedy algorithm, it is not guaranteed that a global optimum will be reached
- Decision tree is very sensitive to changes in data. Small changes in data (i.e. adding samples) can lead to large structural changes in the tree, i.e. high variance
- This is a more complex model and often requires more training time
- Without regularization (early stopping, pruning, max nodes, etc.), there is high risk of overfitting

3.5.3 Random Forest

Random forest is a method that uses ensemble learning to address some of the disadvantages of the decision tree model. Ensemble learning combines results from multiple models to make more accurate predictions than any one single model, by reducing variance. Random forest uses an ensemble learning technique known as bootstrap aggregation, aka bagging. Bagging uses random sampling with replacement to build individual models on subsets of the available data and then aggregate the results into one prediction. The repeated sampling leads to an algorithm that is known to reduce variance, as in one of the main disadvantages of the decision tree model. Random forest combines many decision trees into one model by running the individual decision tree models in parallel and then outputting the prediction that is the mode of target classes for a classification problem or the mean prediction for a regression problem (Chakure, 2020). The structure of a random forest model is shown in Figure 3.3.

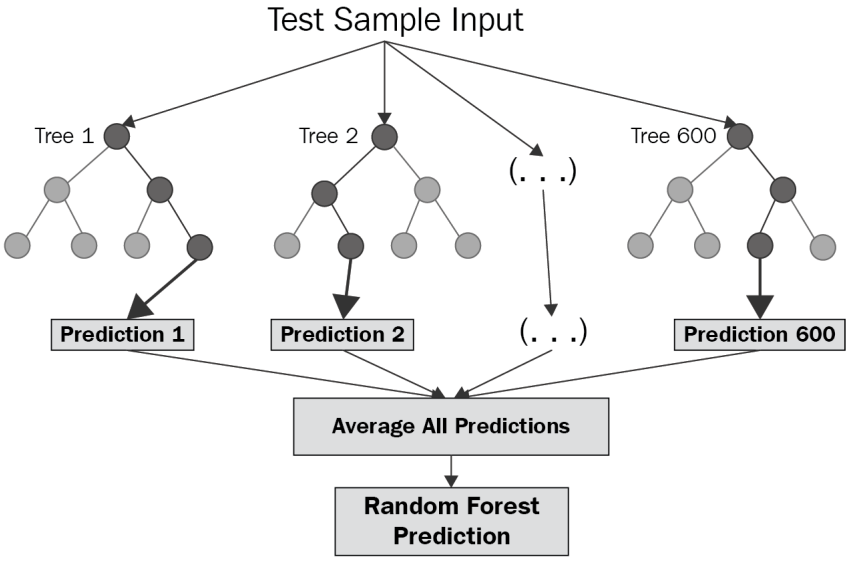


FIGURE 3.3: Random Forest Structure (Chakure, 2020)

Advantages

- Much like decision tree, gives estimates of most important features

- Known for high accuracy, low bias
- Decreased variance in comparison to decision tree
- Can handle large datasets with high dimensionality
- Since it identifies most important features, can be used as a feature reduction method
- Robust to missing data
- Use of bootstrap sampling allows for successful application when data is limited

Disadvantages

- When classifying categorical data, biased in favor of features with more levels
- Will overfit data if regularization not used, such as limiting number of features that can be split at each node
- More difficult to interpret than single decision tree model

3.5.4 Support Vector Machines (SVM)

SVM with Gaussian kernel is a parametric model that represents instances of data as points in space and then builds a model to assign new instances to one category or another. Each data point is represented as a n-dimensional vector, then SVM constructs an n-1-dimensional separating hyperplane to discriminate 2 classes, with maximized distance between the hyperplane and data points on each side. SVM aims to find the best hyperplane for separation of both classes (Rudd, 2018). Data are represented as:

$$(\vec{x}_i, y_i), ..., (\vec{x}_n, y_n) \tag{3.38}$$

where y_i is either 1 or -1, indicating to which class x_i belongs. Each x_i is p-dimensional vector representing all of the characteristic values (features) of x_i. The hyperplane that best separates the group of x_i vectors where $y_i = 1$ from the group of vectors where $y_i = -1$ is:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{3.39}$$

Where $\vec{w}$ is the normal vector to the hyperplane and b is the offset of the hyperplane from the origin. If the data points are linearly separable, the hard margin can be represented as

$$\vec{w} \cdot \vec{x} - b = 1 \tag{3.40}$$

and

$$\vec{w} \cdot \vec{x} - b = -1 \tag{3.41}$$

Figure 3.4 shows a maximum margin separation for linearly separable data. The samples that fall on the margin are known as the support vectors.

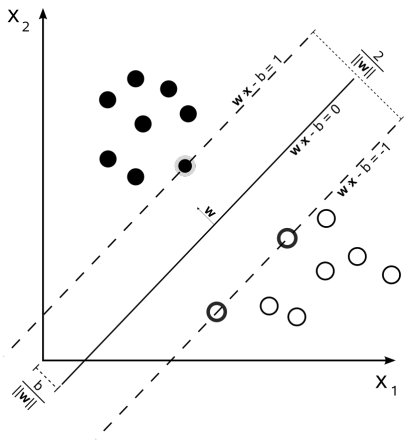


FIGURE 3.4: Maximum Margin Hyperplane (Kumari and Chitra, 2013)

The SVM algorithm assumes that data is in a standard range (usually between 0 to 1, or -1 to 1), so it is recommended to scale features before using the algorithm. In fact, when using the Gaussian kernel, if data is normalized

between 0 and 1, then the dot product between the feature vectors and the separating hyperplane is the cosine similarity (Forman, Scholz, and Rajaram, 2009).

Advantages

- If there is clear separation of the data classes, SVM works very well
- Effective in high-dimensional data, especially when the number of features is similar or greater than the number of samples
- Since the samples that make up the support vectors are the only training data used to define the model, SVM is memory efficient

Disadvantages

- Since this model has to calculate the distance between every training point to create a separating hyperplane, it is computationally expensive as the size of the data set increases
- Noisy data with overlapping target classes are difficult to separate; Kernel functions can be added to transform the data into higher level feature space for improved separation but this adds model complexity
- Does not directly provide parameter coefficients so it is difficult to interpret

3.5.5 Gradient Boosting

Gradient boosting is another form of ensemble learning, this time utilizing a technique known as boosting. In a boosting algorithm, predictions are not made in parallel as in the bagging method of random forest. In this case, subsequent prediction models learn from the mistakes of previous models. Observations have an unequal probability of appearing in the subsequent models, with high error observations appearing in the most models. This is contrary to the random forest model where observations are selected for

each model via bootstrapping (random selection with replacement) and have equal probability of appearing in each model. Visual comparison of single, bagging, and boosting models is show in Figure 3.5.

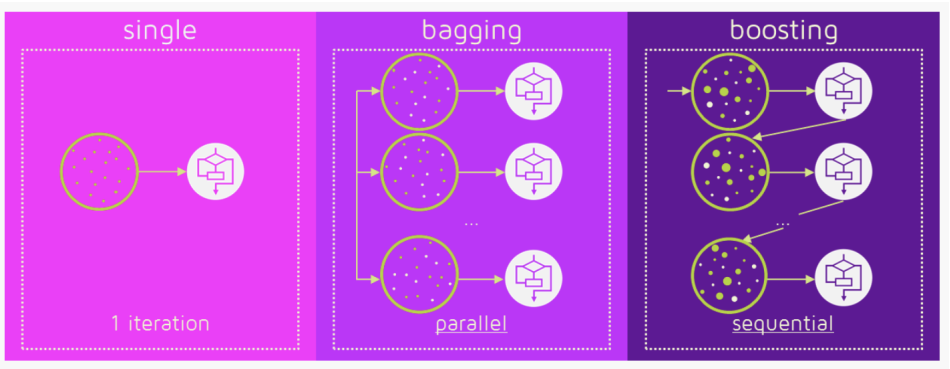


FIGURE 3.5: Bagging (independent models) and Boosting (sequential models) (Grover, 2019)

In gradient boosting, an ensemble of weak models, often decision trees, are used to improve the model based off of hard to predict samples. The algorithm leverages patterns in model residuals, such as those from using MSE loss, to build subsequent models from the weak predictions. For example, in a simple linear regression there is the assumption that the sum of the residuals is 0, i.e. spread randomly with no pattern around zero. However, assuming there is some pattern in the residuals for a base model, such as a decision tree, gradient boosting builds sequential models off of these residual patterns until there is no longer a pattern, i.e. average residual is zero or constant. The sequential model predictions are then weighted into a combined prediction. The intuitive idea behind gradient boosting is to combine several weak models, with each additional weak model improving the MSE of the overall model. Advantages of bagging and boosting ensemble techniques are illustrated in Figure 3.6.

Advantages

- Focus on difficult to classify cases makes it robust to imbalanced datasets

- MSE is commonly used loss function, but gradient boosting can be optimized on many objective functions so it can be extended to many different problem spaces

Disadvantages

- Requires more hyperparameter tuning, and training time to avoid overfitting compared with random forest
- Sensitive to overfitting if data is noisy, i.e. many hard to classify cases to use in the sequential models
- Longer training requirements due to sequential nature of algorithm (as opposed to parallel model development in random forest)

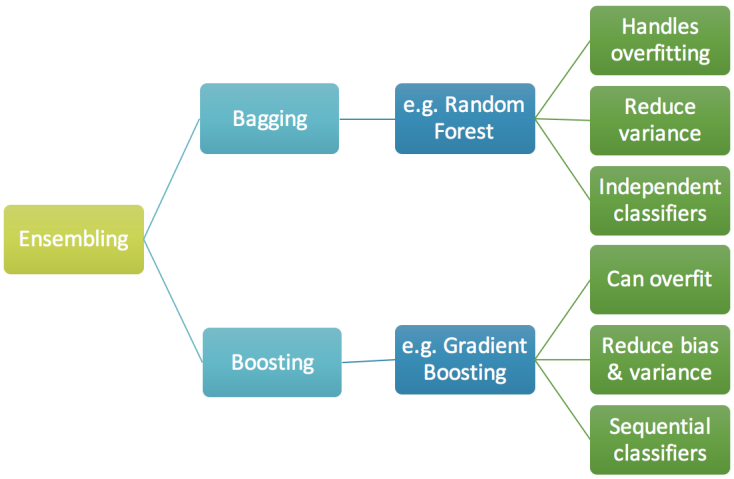


FIGURE 3.6: Ensembling (Grover, 2019)

3.5.6 Neural Network

In this study, the effects of normalization on various data types are also tested using a multi-layer perception (MLP), also known as the simple form of a neural network. Neural networks are models that learn non-linear function approximations by feeding a set of input features into an output. Although

the input and output layers are similar to the linear approximations of generalized linear models, neural networks differ in that there is one or more non-linear hidden layers, as in Figure 3.7 with one hidden layer.

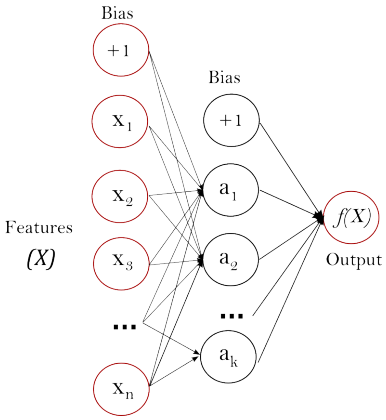


FIGURE 3.7: One hidden layer Neural Network

The first layer, the input layer, contains a set of neurons $x_i|x_1, x_2, ...x_m$ representing the m input features. The inputs are fed into the hidden layer first with a weighted linear combination, similar to the linear combination of features and βs in a GLM. The combined inputs are then transformed by a non-linear activation, such as a *tan* function. From the last hidden layer, the input layer then applies an activation function to transform the values into outputs, such as the *sigmoid* function for a binary classification problem. The weights within each layer of the neural network are learned through a process of backpropagation and gradient descent. This process uses derivatives with respect to each parameter to find the optimal value of the selected loss function. Even though neural network uses non-linear transformations in the hidden layers, the network still uses linear combinations of the features and weights to learn the optimal parameters, as in the GLM, linear-based methods.

Advantages

- Can learn complex, non-linear models

- Works well with "big data"; feeding neural networks more data leads to improved training and results
- Ability to detect all possible interactions between predictor variable

Disadvantages

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum; different random weight initializations can lead to different validation accuracy
- Sensitive to feature scaling, due to above disadvantage
- Requires a lot of tuning (number of hidden neurons, layers, iterations), and regularization to prevent overfitting
- Requires a lot of data for best training and results
- Difficult, computationally expensive to train
- "Black box" algorithm is difficult or not possible to interpret

3.5.7 Model Summary

A global model is one in which there is a single predictive formula for the entire data space. A linear transformation of data in a linear-based global model (linear regression, logistic regression, Poisson regression, linear SVM) will result in the model parameters (i.e. weights in a neural network, coefficients in regression) adjusting to reach the optimal value of the loss function, such as using the MLE in the GLM class of models. As a result, we expect that choice of normalization method should not affect the loss function value as long as the feature space is a convex function, but it can affect the values and stability of the feature coefficients. In this case, even though normalization may not affect the estimated total average error, it may have effects on the estimated average bias and variance due to model instability.

For non-linear, locally recursive models such as decision tree, random forest, and gradient boosting regression, it is also expected that within-feature global normalization will have little effect on loss function value. These tree-based models optimize by finding the best split-point within each individual feature by the percentage of labels correctly classified using that feature. Since these models are local, recursive models, as long as the ordering within the features is preserved, normalization of the data should not affect the loss function value. However, although we're using a decision tree-based learning model for the gradient boosting regression, this type of sequential boosting model relies on minimizing the MSE for the global model through subsequent predictions on the individual model residuals. Because of this, it is suspected that the gradient boosting model will exhibit patterns in bias-variance decomposition similar to the linear models. However, since we are using the default hyperparameters in the gradient boosting model for consistent simulation conditions, it is possible that the bias-variance decomposition results will suffer from overfitting and have a longer training time.

Chapter 4

Methods

4.1 Simulation Methods

In order to approximate the bias-variance decomposition we need to approximate the expected value $\mathbb{E}_\tau[\hat{f}_\tau(X)$ by simulating many variants of the training data sets. We can do this via bootstrap sampling. We take a synthetic input dataset D and create variants of D from $D_1, ..., D_T$ of size n.

Algorithm 1 Bootstrap Sampling

for $t = 1, \ldots, T$ **do**
  $D_t = \emptyset$
  for $i = 1, ..., n$ **do**
    Pick (x, y) uniformly at random from D (i.e., with replacement) and add it to D_t
  end for
end for
Create B bootstrap variants of D
for each bootstrap dataset (b) **do**
  T_b is the dataset and U_b are the "out of bag" examples
  Train a hypothesis $\hat{f}_b$ on T_b
  Test $\hat{f}_b$ on each x in U_b
end for

Now for each (x, y) example we have many predictions $\hat{f}_1(x), \hat{f}_2(x), ..., \hat{f}_B(x)$ and can estimate:

- variance: variance of $\hat{f}_1(x), \hat{f}_2(x), ..., \hat{f}_n(x)$
- bias: average$(\hat{f}_1(x), \hat{f}_2(x), ..., \hat{f}_n(x)) - y$

$B = 1000$ bootstrap replicate datasets with 70% training samples and 30% out-of-bag testing samples were selected from simulated bivariate normal data with $n = 1000$ samples. The simulated features have different means and standard deviations, and an identity covariance matrix. The true target value, Y, was created as a simple linear function of the simulated features plus a random error term. In addition, datasets with binary (logistic) target and continuous target were created for each simulated data distribution. Models of varying complexity were applied using the training data and the bias-variance decomposition of the MSE loss and 0-1 loss computed on the test set, with average loss, bias, and variance calculated over all 1000 bootstrap replicates. In this case, since this was an empirical study approximating model functions by training on repeated bootstrap replicates and testing on the out-of-bag samples, the expected values of the total loss are more specifically referred to as *empirical risk functions*, i.e. *MSE risk* and *0-1 risk*. Models tested included logistic regression, decision tree, random forest, support vector machine (SMV), gradient boosting regression, and neural network. This process was repeated on the simulated data using several normalization techniques including z-score, min-max, "maxAbs" (normalize between -1 and 1), quantile transformation (within feature technique), and quantile normalization (between feature technique commonly used in genetic data normalization). The former technique is a generalization of a commonly used method for normalizing gene sequence read lengths. Initial dataset simulations were completed in R and risk function decompositions with bootstrapping completed in Python. This process was then repeated on additional simulated datasets including rank-based data (similar to sports statistics data), categorical data, mixed data, and Poisson data. MSE and 0-1 risk decomposition were then performed on several benchmark datasets to assess results on various data characteristics including sparse data, wide data (more features than samples), and imbalanced data. These benchmark datasets were from the UCI Machine Learning Library (Dua and

Graff, 2017) and are listed in Table 4.1. Averaged risk function results from simulations were then used to populate the decision points in the proposed framework. Selected model/normalization pairings are determined by selecting the best risk function value, i.e. best MSE and 0-1 risk. The results of bias-variance decomposition are then used as a diagnostic illustration of where researchers can expect to find improvements in their model results, i.e. does risk function improvement result from improved bias, variance, or a combination of both. Finally, the resulting model development framework was applied to several applications, comparing to baseline results from these applications prior to the development of the framework. Selected applications to test the framework include the historical data from the NCAA Men's Basketball Tournament (historical data used due to cancellation of NCAA tournament in 2020; comparing to model results from last years competition), and a credit risk model (Rudd and Priestley, 2017). Application data sources and uses are found in Table 4.1.

Benchmark Datasets					
Dataset	**Target Type**	**Attribute Type**	**Dataset Characteristics**	**# Attributes**	**# Instances**
Wine Quality	Binary	Numeric	Imbalanced	10	4898
Breast Cancer Wisconsin	Binary	Numeric	features have very dissimilar ranges, with half of the features near unary at 0	30	569
Congressional Voting Records	Binary	Categorical	Missing data	16	435
Abalone	Binary	Mixed	Imbalanced	8	4177
Arrhythmia	Binary	Mixed	Imbalanced; small dataset; # features more than 1/2 # of instances	279	452
Forest Fires	Continuous	Numeric	No missings	13	517
Solar Flare	Continuous	Categorical	# of common solar flares within 24h; distribution of target is highly skewed towards 1	10	1066
Auto MPG	Continuous	Mixed	No missings	8	398

TABLE 4.1: Benchmark Datasets and Charactertics

Application Datasets		
Application	**Dataset(s)**	**Notes**
NCAA tournament	Data from 1985 to present available on Kaggle.com; additional metrics available for public use from sports statistics sites	2020 tournament canceled. Will test on 2018 tournament data instead
Credit Risk Model	Equifax data available from Binary Classification Course	Previous results had logistic regression performing slightly better than decision tree. Re-do analysis and select best data normalization methods for logistic and decision tree based on findings from proposed model development framework

FIGURE 4.1: Application Datasets

Chapter 5

Experimental Results

This chapter is divided into three sections describing results from 1) bias-variance decomposition simulation, 2) bias-variance decomposition on benchmark datasets, and 3) application of findings from sections 1 and 2 to existing NCAA data and credit risk data. The first section on simulation results is divided by target data type (binary, continuous, Poisson), with details of results provided by feature data structure and model. In this case, specific details of results are provided for all models applied to bivariate normal data structure, with additional summaries of results provided for the other considered data structures (ranked, categorical, mixed data). Results across data structures and models is relatively consistent so summaries were provided to avoid repetition. Complete results of bias-variance decomposition under various data structures, normalization strategies, and models are shown in Appendices A and B. Performance measures for simulated model results are found in Table 5.1.

5.1 Simulation Results

5.1.1 Binary Target

Logistic Regression Results

Based on results from simulated bivariate normal data with a linearly dependent binary target (Table A.1 and Figure B.1), the best performing normalization,

quantile normalization, does not significantly improve empirical risk as compared with the raw data when using logistic regression, with both methods resulting in risk of 0.290. The within-feature normalization choices (z-standardization, min-max, maxAbs, and quantile transformation) result in worse performance in the logistic regression model due to increased average variance even though bias is equal or improved from the raw and quantile normalized data (Figure B.1a), with MSE and 0-1 risk ranging from 0.462 for z-standardization to 0.485 for maxAbs. Use of these methods results in variance to bias ratio ranging from 1.06 for MSE risk under quantile transformation to 1.9 for MSE risk under MaxAbs normalization. In addition, 0-1 risk estimates for the within-feature normalization methods have increased noise due to the non-additive nature of the 0-1 risk decomposition, whereas the MSE risk estimates average zero additional noise.

Finally, use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (0.190), with bias (0.237) plus variance (0.438) greater than the average total error (0.485). This effect is seen in the decomposition results of all the simulated data and target types. This is potentially due to the extreme value distribution problem, where the extreme values in the original distributions result in maxAbs normalized data squished around a small point. As a result, this normalization is not recommended for data with large values or outliers.

Decision Tree Results

Using a decision tree model on the simulated bivariate normal data with binary target results in increased risk function values across all normalizations and raw data as compared with the logistic regression model, with risk ranging from 0.393 using quantile normalization to 0.564 using any of the within-feature normalization strategies (B.1b). The decision tree model results in increased bias and variance across all normalizations in comparison with logistic regression.

The within-feature normalization methods have the worst overall performance for this model, particularly among 0-1 risk decomposition where the methods result in more than doubling of the bias (0.71) as compared with the other methods (0.35). The decision tree risk function values across all methods are driven by high bias, as illustrated by variance to bias ratios ranging from 0.49 for 0-1 risk under both z-standardization and quantile transformation to 1.03 for MSE risk under maxAbs normalization. As in the logistic regression model results, decision tree results illustrate the non-additive effect of 0-1 risk with increased noise (0.174 to 0.490), as well as the unstable MSE results under the maxAbs normalization method.

Random Forest Results

Random forest model results on this data result in similar risk function behavior as in the logistic regression model, with raw data and quantile normalization both having best risk function value of 0.295 (B.1c). Both methods also result in low variance between 0.009 for MSE risk and 0.011 for 0-1 risk, with variance to bias ratio between 0.032 for MSE and 0.038 for 0-1 risk. The within-feature normalization methods perform worse due to increased variance between 0.188 under MSE risk for z-standardization, min-max, and quantile transformation to 0.252 under 0-1 risk for the same methods. Random forest model results also illustrate the same non-additive behavior of the 0-1 risk decomposition, as well as the unstable MSE results under the maxAbs normalization method.

Support Vector Machine (SVM) Results

Risk function results using a support vector machine (SVM) model with Gaussian kernel, are normalization-agnostic, with raw and quantile normalized data having risk of 0.292 and all other methods with risk of 0.290 (B.1d). The results are consistent regardless of risk function or normalization method

use, and do not exhibit the same irregularities in the 0-1 risk decomposition or maxAbs methods found using the other models.

Gradient Boosting Results

Using a gradient boosting model on the simulated bivariate normal data with binary target results in increased risk function values across all normalizations and raw data as compared with the logistic regression, decision tree, and SVM models, with risk ranging from 0.332 using raw data and quantile normalization to 0.655 using any of the within-sample normalization strategies (B.1e). The within-feature normalization methods have the worst overall performance for this model with bias more than doubling as compared to the other methods. The gradient boosting risk function values across all methods are driven by high bias, as illustrated by variance to bias ratios ranging from 0.186 for 0-1 risk under z-standardization, min-max, maxAbs, and quantile transformation to 0.38 for 0-1 risk using raw data. As in the logistic regression, decision tree, and random forest model results, gradient boosting results illustrate the non-additive effect of 0-1 risk with increased noise (between 0.073 and 0.187), as well as the unstable MSE results under the maxAbs normalization method, with average noise of 0.018.

Neural Network Results

As in the SVM model results, choice of normalization has little to no effect on resulting risk function value using a neural network, with total average risk ranging between 0.381 for 0-1 risk using quantile normalized data to 0.427 for MSE risk using z-standardized data (B.1f). As opposed to the SVM model however, the risk function bias-variance decomposition varies slightly within each normalization method using the neural network. For example, while neural network results have lower bias across all normalization methods in comparison to the SVM model, the risk decomposition has increased variance

compared to SVM, resulting in higher total average risk. Once again, these model results illustrate the non-additive effect of 0-1 risk with increased noise (between 0.124 and 0.185), as well as the unstable MSE results under the maxAbs normalization method, with MSE average noise of 0.093.

Generalized Results

Over all models applied to bivariate normal data with binary target, SVM using within-feature normalization methods (risk = 0.290), and logistic regression with raw data or quantile normalization (0.290) have best, similar risk function results (A.1). While SVM has the consistently best results and is normalization-agnostic, logistic regression with raw data or quantile normalization has similar performance with a faster run time (approximately 5 seconds vs 18 seconds as in Table 5.1). If an analyst would like to use decision tree, random forest, or neural network models instead, it is recommended to use raw data or quantile normalization for best risk function results.

For models applied to rank-based data with binary target, logistic regression using raw data or quantile normalization (risk = 0.444), and SVM with raw data or quantile normalization (0.437) have best, similar risk function results (A.4). While SVM has the consistently best results, logistic regression with raw data or quantile normalization has similar performance with a faster run time (approximately 6 seconds vs 39 seconds as in Table 5.1). If an analyst would like to use decision tree (best risk = 0.489), random forest (best risk = 0.445), gradient boosting (best risk = 0.465), or neural network (best risk = 0.475) models instead, it is recommended to use raw data or quantile normalization for best risk function results.

Over all normalization methods and models applied to categorical data with binary target, risk function ranged from 0.473 to 0.502, indicating that this data is somewhat model- and normalization-agnostic (A.7). SVM using z-standardized data, and logistic regression with all methods except z-standardization

resulted in best risk function value of 0.473. However, while SVM and logistic regression have similar results, logistic regression is more than 3 times faster (7 seconds vs. 21 seconds average processing time as in Table 5.1). Since the simulated dataset consists of all categorical data, the features are first converted to [0,1] coded dummy features, effectively "normalizing" the data between 0 and 1, so additional normalization methods are not expected to have an effect on the downstream analysis.

For mixed data types with a binary target, although there are slight deviations between normalization and model performance, risk function values do not vary much between all methods, with a range between 0.479 and 0.507 (A.10). A decision tree model using raw data leads to the best results, and gradient boosting using raw or quantile normalized data leads to the worst results. However, considering the consistency of performance across normalization methods and models, it is recommended to make selections based on additional criteria, such as processing resources, model interpretation, or another performance measure such as specificity and sensitivity.

5.1.2 Continuous Target

Linear Regression Results

Based on results from simulated bivariate normal data with a linearly dependent continuous target (Table A.2 and Figure B.2), the best performing normalization, quantile normalization, does not improve empirical risk or bias-variance decomposition of the risk as compared with the raw data when using linear regression, with raw data resulting in risk of 0.338 and quantile normalization resulting in risk of 0.344. The within-feature normalization choices result in worse performance in the linear regression model due to increased average bias and variance (Figure B.2a), with MSE ranging from 2×10^5 for z-standardization to 9×10^6 for maxAbs. Use of these methods results in variance to bias ratio

ranging from near 0 for MSE risk under the within-feature normalizations to 0.019 for MSE risk under quantile normalization. In all cases, bias significantly outweighs variance in total risk estimate.

Finally, use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (3022.5), with bias (2,146,478,729.7) plus variance (12.3) greater than the average total error (2,146,481,764.5).

Decision Tree Results

Using a decision tree model on the simulated bivariate normal data with continuous target results in increased risk function values compared with the best performing raw and quantile normalized data in the linear regression model, but improved risk compared to the within-feature normalizations. Risk ranges from 0.589 using quantile normalization to 114.6 using any of the within-feature normalization strategies (B.2b). The decision tree risk function values across the within-feature methods are driven by high bias, as illustrated by variance to bias ratios of 0.24, and bias that is approximately 250 times higher than that in the raw and quantile normalized data.

Random Forest Results

Random forest model results on this data result in similar risk function behavior as in the decision tree model, with raw data and quantile normalization both having best risk function value of 2.8 (B.2c). However, the best methods still result in worse risk function performance than the best methods found in any of the other tested models. The best risk function values range between 0.338 to 0.65 for all other tested models. The within-feature normalization methods perform worse due to increased bias (22.9) approximately 9 times higher than that found in raw and quantile normalized data (2.4).

Support Vector Machine (SVM) Results

Risk function results using a support vector machine (SVM) model with Gaussian kernel, result in similar risk function behavior as in the decision tree and random forest models, with raw data and quantile normalization both having best risk function value of approximately 0.65 (B.2d). However, risk function performance across all methods is improved compared to decision tree and random forest. The SVM risk function values across the within-feature methods are driven by high bias, as illustrated by variance to bias ratios of 0.009 to 0.013, and bias that is approximately 14 times higher than that in the raw and quantile normalized data. Finally, use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (0.019), with bias (8.75) plus variance (0.112) greater than the average total error (8.84).

Gradient Boosting Results

Using a gradient boosting model on the simulated bivariate normal data with continuous target results in risk function values ranging from 0.55 using raw data and quantile normalization to approximately 152 using any of the within-feature normalization strategies (B.2e). The within-feature normalization methods have the worst overall performance for this model with bias increasing 520-fold compared to the other methods. The gradient boosting risk function values across the within-feature methods are driven by high bias, as illustrated by variance to bias ratio of 0.007 for MSE risk. In the raw data and quantile normalized data methods, bias and variance represent nearly equal weight in the risk function decomposition. As in the linear regression and SVM model results, gradient boosting results illustrate the unstable MSE results under the maxAbs normalization method, with average noise of 0.001.

Neural Network Results

Based on results from simulated bivariate normal data with a linearly dependent continuous target, the best performing normalization, quantile normalization, does not improve empirical risk or bias-variance decomposition of the risk as compared with the raw data when using the neural network model, with raw data resulting in risk of 0.341 and quantile normalization resulting in risk of 0.356. (B.2f). The within-feature normalization choices result in worse performance in the neural network model due to increased average bias and variance, with MSE ranging from 200,151.85 for z-standardization to 2,145,420,994.19 for maxAbs. Use of these methods results in variance to bias ratio ranging from near 0 for MSE risk under the within-feature normalizations to 0.132 for MSE risk under quantile normalization. In all cases, bias significantly outweighs variance in total risk estimate.

Finally, use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (123,606), with bias (2,145,297,375.91) plus variance (12.6) less than the average total error (2,145,420,994.19).

Generalized Results

Over all models applied to bivariate normal data with continuous target, neural network using raw data (risk = 0.342), and linear regression with raw data (0.338) have best, similar risk function results (A.2). Quantile normalization method for both models has similar results with MSE risk of 0.344 for linear regression and 0.356 for neural network. However, while neural network and linear regression have similar results, linear regression is approximately 20 times faster (1.2 seconds vs. 26 seconds average processing time as in Table 5.1). SVM has the most consistent results; even though the within-feature normalization methods all perform worse than raw or quantile normalized data, SVM within-feature normalization results perform better than the same

normalization in all other tested models. If normalization and scaling of data is required, as with features measured on highly divergent scales, it is recommended for an analyst to test the SVM model, keeping in mind increased processing requirements.

When applied to rank-based data with continuous target, neural network using raw and quantile normalized data (risk = 0.175), and linear regression with raw and quantile normalized data (0.174) have best, similar risk function results (A.5). However, while neural network and linear regression have similar results, linear regression is more than 17 times faster (0.8 seconds vs. 14 seconds average processing time as in Table 5.1). SVM has the most consistent results; even though the within-feature normalization methods all perform worse than raw or quantile normalized data, SVM within-feature normalization results perform better than the same normalization in all other tested models. If normalization and scaling of data is required, as with features measured on highly divergent scales, it is recommended for an analyst to test the SVM model, keeping in mind increased processing requirements. Note, however, that outside of the best-performing linear regression and neural network models, all other methods and models perform significantly worse due to exploding estimates of average bias.

While normalization generally does not improve or worsen results as compared with raw data, it is not recommended to use z-standardization for categorical data with a continuous target, as the simulation results indicate increased risk function values (A.8). In particular, if using linear regression, z-standardization and quantile transformation should be avoided as these methods used with this model lead to significant explosion in the risk function value. Outside of z-standardization for any tested model, and quantile transformation for linear regression, simulation results indicate that this type of data is both normalization- and model-agnostic. In this case, normalization and model selection can be based off of additional criteria, such as processing requirements

or model transparency.

Over all models applied to mixed data with continuous target, neural network using raw data (risk = 0.366) and quantile normalized data (risk = 0.425), and linear regression using raw data (risk = 0.367) and quantile normalized data (0.363) have best, similar risk function results (A.11). However, while neural network and linear regression have similar results, linear regression is approximately 41 times faster (1.6 seconds vs. 66 seconds average processing time as in Table 5.1). SVM has the most consistent results; even though the within-feature normalization methods all perform worse than raw or quantile normalized data, SVM within-feature normalization results perform better than the same normalization in all other tested models. If normalization and scaling of data is required, as with features measured on highly divergent scales, it is recommended for an analyst to test the SVM model, keeping in mind increased processing requirements and potential for increased bias.

5.1.3 Poisson Target

Poisson Regression Results

Based on results from simulated bivariate normal data with a Poisson process target (Table A.3 and Figure B.3), the best performing normalization, quantile normalization, does not improve empirical risk or bias-variance decomposition of the risk as compared with the raw data when using Poisson regression, with both raw data and quantile normalization resulting in risk of 1.502. The within-feature normalization choices result in worse performance in the Poisson regression model due to increased average bias or variance, depending on the method used (Figure B.3a). For example, using z-standardization results in variance increasing more than 700-fold, and use of quantile normalization leads to a 7-fold increase in variance as compared to the raw data.

Finally, use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (0.101), with bias (1.6) plus variance (0.264) greater than the average total error (1.764).

Decision Tree Results

Using a decision tree model on the simulated bivariate normal data with poisson process target results in increased risk function values compared with all methods and all models, except for neural network. Risk ranges from 1.8 using within-feature normalizations to 2.065 using quantile normalization (B.3b). In this case, the decision tree risk function values are improved by using within-feature normalization strategies. Increased risk function value for raw data and quantile normalized data are due mostly due increased variance. In fact, even though bias increases when using the within-feature strategies, the 2-fold decrease in variance across these methods results in lower empirical risk function values.

Random Forest Results

Random forest model results on this data result in similar risk function behavior as in the decision tree model, with within-feature methods having best risk function value of 1.125 (B.3c). However, in this case, improved risk function results are due to both deceased bias and variance, with bias improving by 29% and variance improving by 26%. These risk function results are the best of all tested methods.

Support Vector Machine (SVM) Results

Using a support vector machine (SVM) model with Gaussian kernel, the best risk function result (1.324) is found using z-standardization (B.3d). Z-standardization, min-max, and quantile transformation all perform better than raw data, maxAbs,

and quantile normalization due to improved bias. For example, even though variance increases from 0.148 to 0.233, z-standardization improves bias to 1.091 from 1.528 when compared with raw data, leading to better overall performance. Finally, use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (0.171), with bias (1.805) plus variance (0.252) greater than the average total error (1.886).

Gradient Boosting Results

Using a gradient boosting model on the simulated bivariate normal data with Poisson target results in risk function characteristics similar to those illustrated in decision tree and random forest models (B.3e). The within-feature normalization methods have the best overall performance for this model with risk function value of 1.167. As in the random forest model, improved risk function results are due to both decreased bias and variance, with bias improving by 29% and variance improving by 5%.

Neural Network Results

Based on results from simulated bivariate normal data with a Poisson target, the best performing normalization, quantile normalization, slightly improves empirical risk as compared with the raw data when using the neural network model, with raw data resulting in risk of 1.501 and quantile normalization resulting in risk of 1.496. (B.3f). The within-feature normalization choices result in worse performance in the neural network model due to increased average bias and variance, with MSE ranging from 50.24 for z-standardization to 5.5×10^5 for maxAbs. Use of these methods results in variance to bias ratio ranging from near 0 for MSE risk using maxAbs to 1.029 for MSE risk under min-max normalization. Use of the maxAbs normalization technique is the only time the MSE decomposition has additional average noise (294,525.4),

with bias (257,374.38) plus variance (1324.9) less than the average total error (553,224.67).

Generalized Results

Over all models applied to bivariate normal data with Poisson target, random forest using within-feature normalization methods (risk = 1.125), and gradient boosting using within-feature normalization methods (1.167) have best, similar risk function results (A.3). Poisson regression using raw or quantile normalized data also has strong results with risk function value of 1.5. Although Poisson regression results are not as strong as those found using random forest and gradient boosting, Poisson regression has processing time more than 200 times faster than random forest (0.8 seconds vs. 167 seconds average processing time) and 13 times faster than gradient boosting (0.8 vs. 167 seconds average processing time) as seen in Table 5.1.

For rank-based data with Poisson target, SVM using raw and quantile normalized data (risk = 1.281), and Poisson regression with raw and quantile normalized data (1.280) have best, similar risk function results (A.6). However, while SVM and Poisson regression have similar results, Poisson regression is more than 6 times faster (6 seconds vs. 36.6 seconds average processing time), as seen in Table 5.1. Although Poisson regression has the best results for this data, use of the within-feature normalization methods result in unstable, exploding risk function values and should be avoided. Within-feature normalizations should also be avoided if using a neural network on this data for the same reason. If normalization and scaling of data is required, random forest model has the best results for the within-feature methods, although it has the longest processing time.

Although there are slight deviations between normalization and model performance, risk function values do not vary much between all methods when considering categorical data with Poisson target, with a range between

1.499 and 1.783 (A.9). A decision tree model using min-max, maxAbs, quantile transformation, or quantile normalization lead to the best results, and SVM using z-standardization leads to the worst result. However, considering the consistency of performance across normalization methods and models, it is recommended to make selections based on additional criteria, such as client requested models.

Over all models applied to mixed data with Poisson target, gradient boosting using raw and quantile normalized data (risk = 1.746), and random forest using raw data (1.820) have best results (A.12). Generally, the best performing risk function values do not vary much between all tested models, with a range between 1.476 for the gradient boosting model and 2.087 for the decision tree model (Table 5.1). The similarity in best performing model results indicates that this data type is somewhat model-agnostic, although normalization methods should be selected carefully if required for analysis. For example, it is not recommended to use any of the tested within-feature normalizations if a neural network is used due to significant increases in bias and variance found in the simulation results.

	Features	Bivariate Normal	Ranked	Categorical	Mixed
Target					
Binary	**Model**				
	Logistic Regression	5	6	7	14.2
	Decision Tree	2	3	15	3.3
	Random Forest	139	209	183	208.6
	SVM	18	39	21	37.3
	Gradient Boosting	14	25	14	22.7
	Neural Network	223	356	262	287
Continuous	**Model**				
	Linear Regression	1.2	0.8	8.8	1.6
	Decision Tree	3	2.3	1	2.8
	Random Forest	206	167	156.9	170.9
	SVM	30	42	10	30.4
	Gradient Boosting	17	13	8.7	10.2
	Neural Network	26	14	6.8	66.4
Poisson	**Model**				
	Poisson Regression	0.8	6	2.4	8
	Decision Tree	2.3	2.7	1	8
	Random Forest	167	173.7	158.4	3.4
	SVM	42	36.6	22.2	176.5
	Gradient Boosting	13	12.9	7.5	9.5
	Neural Network	14	24	6.6	43.1

TABLE 5.1: Average model performance (in seconds) for bias-variance decomposition of simulated data structures.

5.1.4 Framework

The results discussed in the previous sections indicate that the choice of normalization under various data conditions and models do affect predictive model risk functions and should be considered when making model selections under certain situations. Results from simulations are represented as a heatmap in a blown up section of the proposed model development framework in Figure 5.1, where best-performing normalization methods for each data type are highlighted in green.

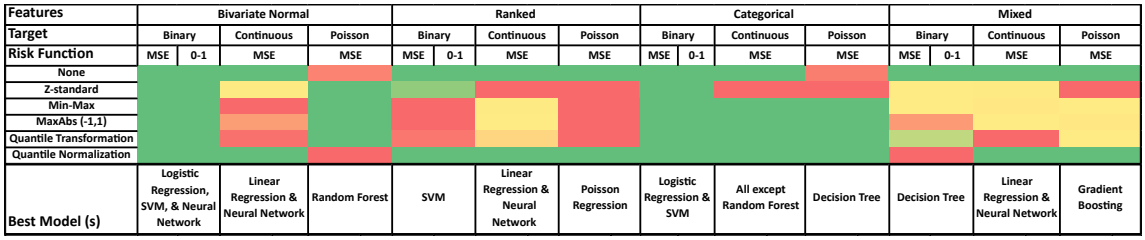


FIGURE 5.1: Heatmap representing empirical risk function values as a percentage of the best performing normalization strategy for each data type, with green as best performing strategies.

5.2 Benchmark Data Results

Benchmark datasets were selected from the UCI Machine Learning Library to cover data types similar to those covered in the simulations. Complete tables of risk function decomposition results are found in Appendix Section A.0.2, and figures are found in Appendix Section B.0.2. Binary target datasets with numeric features (wine quality, breast cancer), and categorical features (congressional voting records), have bootstrapped bias-variance decomposition results consistent with those found in the simulated datasets with the same data structure characteristics (see Figures B.13, B.14, and B.15). The traditional within-feature normalization methods (z-standardization, min-max, maxAbs, quantile transformation) result in risk function values that are the same or worse than using raw data or quantile normalization. For the wine quality data, using raw or quantile normalized data in logistic regression, linear SVM, or neural network results in best risk function performance, while quantile normalization with logistic regression was best for the breast cancer data. For the congressional voting records data, logistic regression and neural network with raw and quantile normalized data were also found to be the best method-model combinations, consistent with simulated data results. However, it is interesting to note that z-standardization in both of these models resulted in the worst risk function performance among all other method-model combinations applied to this dataset, due to both increased bias and variance.

For the binary target data with mixed data type features (arrhythmia, abalone), raw data and quantile normalization also lead to the best risk function performance. However, the arrhythmia dataset is a relatively more complex dataset in comparison to the others tested. It has missing data, many features in comparison to few instances (i.e. 279 features vs. 452 instances), and imbalanced target data. As a result, logistic regression is not well suited for describing these complex relationships, and has worse risk function performance; decision tree and gradient boosting regression with raw data or quantile normalization have best results (Figure B.17). In contrast, while the abalone dataset is also an imbalanced dataset, it has less complex data structures with only 8 features and over 4000 instances. In this case, logistic regression, linear support vector machine, and neural network are well-suited for the less complex data, and result in improved risk function values due to decreased variance in comparison to the more complex models (Figure B.16).

For assessing results on continuous target data, the forest fires dataset (numeric features), solar flare dataset (categorical features), and auto MPG dataset (mixed type features) were considered. Once again, in all cases, the within-feature normalization performed the same or worse than using raw data, with the between-feature quantile normalization process being the only method that resulted in same or some improvement to risk function values. For both numeric and categorical only datasets (forest fires and solar flare, respectively), the linear-based logistic regression and neural network models with raw data or quantile normalization resulted in best performance (see Figures B.18 and B.19), with all other normalization-model combinations resulting in both increased bias and variance. In the case of the more complex mixed feature type auto MPG dataset, gradient boosting regression also has improved performance, but logistic regression has similar performance and is a faster algorithm (Figure B.20).

5.3 Applications

To test the utility of the proposed model development framework, best-performing normalization-model combinations were applied to previous research, and results with and without use of the framework compared.

5.3.1 NCAA Tournament Data

For the 2019 NCAA Men's Basketball Tournament Bracket prediction problem, data was used from over 100,000 NCAA regular season games, with the goal to take information about two teams as input, and output a probability of team 1 winning a game. Motivated by the popular Kaggle competition, models were developed to minimize log-loss between predicted win probabilities and actual game outcomes, as in:

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(y_{\hat{i}}) + (1 - y_i log(1 - y_{\hat{i}})] \tag{5.1}$$

This loss function has high penalty for models that are both confident and wrong (Yuan et al., 2015). Model development involved:

- Readily available game statistics, provided by Kaggle
- Commonly used external ratings systems (Massey Ratings)
- No additional feature engineering
- No domain knowledge

The analysis value-added in comparison to previous research and public models found on Kaggle was to focus on the comparison of various normalization techniques for model development. In particular, the use of outside domain knowledge (public health, genetics) to apply a technique from one domain (genetic research) to an unrelated domain (sports data) proved advantageous

but was not, initially, statistically motivated. However, through simulation of bias-variance decomposition and findings from application to benchmark data, it is expected that improved loss function performance for this type of data (ranked data, balanced target, non-missing data) can be achieved by using a linear-based model with raw or quantile normalized data. Rather than iterating through many normalization-model combinations (which took over 12 hours of computation time when building the model for the 2019 tournament), logistic and linear SVM models with raw and quantile normalized data were trained on NCAA regular season data from 2014-2017 and tested on the 2018 tournament. The same features were used as in the 2019 model, with only the model and normalization selection process updated based on the model development framework findings. From the simulation results, logistic regression and SVM for both raw and quantile normalization provided similar results, although SVM outperformed logistic somewhat due to decreased variance, although it has increased bias. In the updated NCAA application on 2018 data, logistic regression with raw data outperformed the other tested models, with a log-loss score of 0.569 (Figure 5.2). This log-loss score, in comparison to other Kaggle submissions in the 2018 tournament, would have ranked 23rd out of 933 teams (98 percentile) and required only the original data supplied by Kaggle and no additional feature engineering or model tuning. In addition, the entire model development process and testing took less than 30 minutes. Three out of the four models developed correctly predicted the Final Four including the tournament Champion, Villanova. This is compared with a 2019 bracket that, while it correctly predicted the tournament winner, only predicted two of the Final Four teams, and scored in the 90th percentile of Kaggle Log-loss scores.

FIGURE 5.2: 2018 Men's Basketball March Madness bracket developed using winning model, logistic regression with raw data.

5.3.2 Credit Risk Data

Previous research by Rudd and Priestley (2017) compare the use of logistic regression and decision trees for prediction of commercial credit risk. The dataset, provided by Equifax, included over 11 million records and over 300 features, and involved extensive data preprocesing including imputation, feature reduction, and transformation. The effects of normalization were not considered at the time. Based on findings from simulations and benchmark results, it was found that gradient boosting regression with raw and quantile normalization should also be considered for this type of data. Running the analysis again, this time including gradient boosting regression, found best results for gradient boosting with raw data (AUC = 0.96). A drawback, however, of this result in the context of credit risk analysis is that gradient

boosting is much more difficult to explain than the logistic regression and decision tree models, and can be problematic in a heavily regulated industry where model interpretability is required.

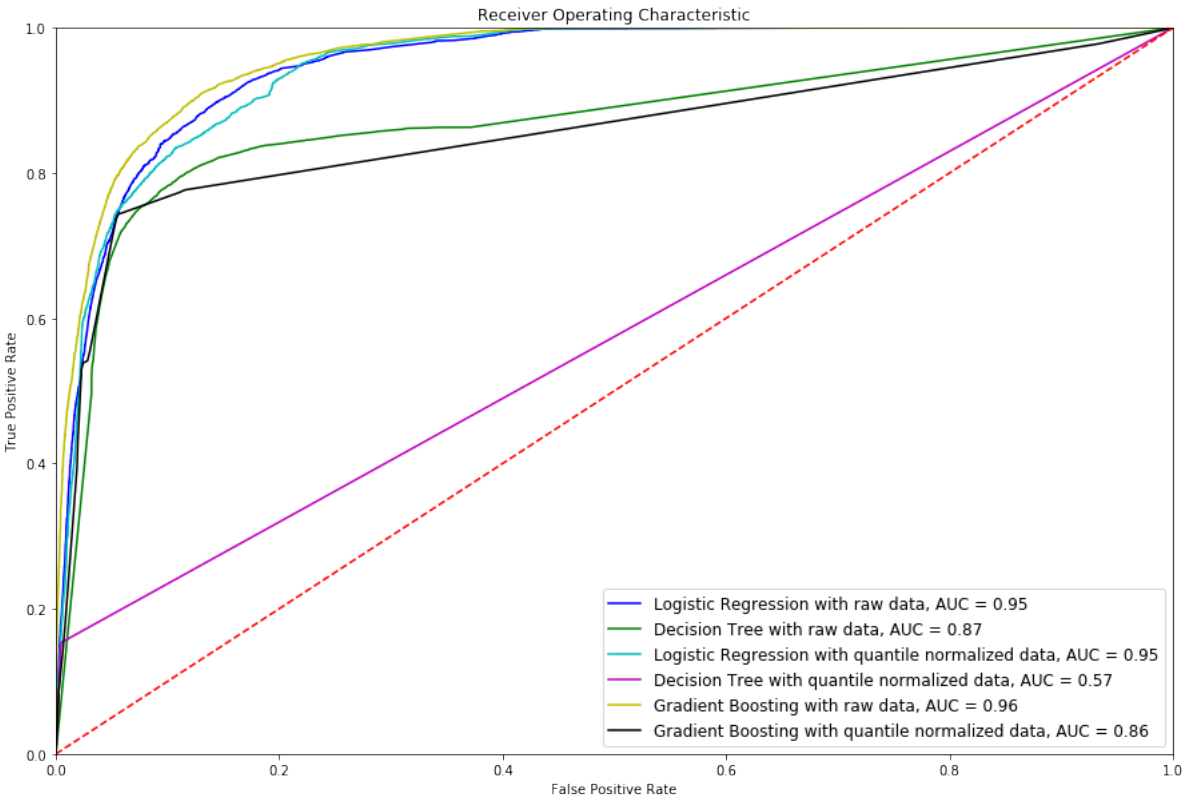


FIGURE 5.3: AUC Curves for selected model-normalization combinations tested based on model framework results

Chapter 6

Discussion

6.1 Conclusions

The goal of this research is to propose a unified model development framework that allows researchers to make statistically motivated variable preparation and model selection choices within the model development pipeline. In fields of social science, theoretical frameworks are common, and often required elements of academic research. However, the current state of data science suffers from an inconsistent approach to modeling strategies, with the complexities of big data, and potentially bad data, inadequately addressed. Review of current data science research and proposed epistemology found that:

- Even "big data" does not exist in a vacuum
- What constitutes "big data" is not consistent across domains
- No Free Lunch (NFL) Theorem: no single algorithm is better than all others on all problems
- Choice of analytic strategy, whether statistically valid or not, has effects on the results

Perhaps the most illustrative lack of research consistency is the study by Silberzahn, et. al. (2018) which recruited 29 independent research teams with 61 analysts to address the question, “Are soccer referees more likely to give

red cards to dark-skin-toned players than to light-skin-toned players?" The research teams represented 13 countries, a variety of disciplines, and a range of expertise and academic degrees. Using the same dataset and research question, the 29 teams utilized 29 unique analytical modeling approaches resulting in 21 unique combinations of covariates, 20 teams with significant positive results, and odds ratios ranging from 0.89 to 2.93, as in Figure 2.4. Analytic choices, even if justifiable and statistically valid, have a downstream effect on model results. There appears to be no unified, quantitatively motivated model development framework for making these analytic choices.

The model development framework can be generally divided into three phases: data discovery, variable preparation, and modeling. Within each of these phases there are steps in the model development that encompass a wide range of data management, data mining, and data analysis techniques, including data ingestion, sample selection, data cleaning and imputation, feature reduction, feature engineering, normalization, model development, and model validation. Analyzing the downstream effects of modeling approaches within each of these steps should be an important goal of the data science community in order to make better informed, statistically motivated modeling choices in the future. Understanding and further analyzing the downstream effects of model development strategies is an important step towards a unified model development framework in the field. Quantifying the analysis effects of these strategies in a unified framework provides a diagnostic illustration of where researchers can expect to find improvements in their model results. The general idea of the proposed framework is that researchers can select analysis methods based on the understanding that model results are a function of the selected model, the selected model development strategies, and the characteristics of the data:

$$L = f(M, [p_1, ...p_i], D) \tag{6.1}$$

where L is the loss function value, M is the selected model, $p_1 ... p_i$ is the list of model development strategies used, and D is the data structure.

Additionally, a simulation study was used to determine empirical justification of the proposed framework. This study tests the utility of the proposed framework by investigating the effects of normalization on downstream analysis results. Normalization methods are investigated by utilizing a decomposition of the empirical risk functions, measuring effects on model bias, variance, and irreducible error. Estimates of bias and variance are then used as diagnostic procedures for data pre-processing and model development. The use of bias-variance decomposition as a unified model framework diagnostic extends from the work proposed by Dietterich and Kong (1995) that "an important goal in algorithm design is to minimize statistical bias and variance and thereby minimize error," and that "any change that increases the representational power of an algorithm can reduce its statistical (and ML) bias. Any change that expands the set of available alternatives for an algorithm or makes them depend on a smaller fraction of the training data can increase the variance of the algorithm." We use our findings to propose model development and algorithm design choices that best minimize common design effects on bias and variance. The result of such a study is to formulate a theory of bias and variance reduction and predict when either or both will succeed in practice.

Both the traditional statistical risk function of mean square error (MSE) as well as the common machine learning risk function of misclassification (0-1 loss) are considered, and the effects of a selection of normalization methods are measured on both risk functions where appropriate. Normalization techniques are selected that represent both data invariant as well as data variant normalization strategies. For example, techniques such as z-score standardization (transforms data to have a mean of zero and a standard deviation of 1) and feature scaling (rescaling data to have values between 0 and 1) change the spread and position of data points (all by consistent factors) but do not change the distribution

shape of the data, whereas techniques such as quantile normalization, commonly used in genetic differential expression analysis, affect the measures of spread, position, and shape. Through simulation of various data structures and bootstrap sampling of the two considered definitions of bias-variance decomposition, a heatmap of best performing model-normalization-data structure combinations was developed to illustrate the empirical justification of an aspect of the proposed unified model development framework. For example, it was found that for rank-based data with binary target, quantile normalization performed better than the data invariant methods with similar or improved performance over raw data due to decreased variance in the loss function value. In addition, results found from simulations were verified and expanded to include additional data characteristics (imbalanced, sparse) by testing on benchmark datasets available from the UCI Machine Learning Library. Normalization results on benchmark data are consistent with those found using simulations, while also illustrating that more complex and/or non-linear models provide better performance on datasets with additional complexities, such as wide data (large feature to instance ratio) as in the arrhythmia dataset. Finally, applying the model development framework and findings from simulation experiments to previous applications led to equivalent or improved results with less model development overhead and processing time. Applying the model framework to the 2018 NCAA Men's Basketball data resulted in a log-loss score that would have been ranked 23 out of 933 teams (98th percentile) and only required 30 minutes of model overhead, as opposed to a 2019 model that required over 12 hours of processing and resulted in a 90th percentile log-loss score.

6.1.1 Limitations

While this work establishes a justification for a unified model development framework in data science, the statistical illustration of the downstream effects

of such a framework is limited in scope, and represents a baseline for further research in this area. For example, while the bias-variance decomposition simulations described in this dissertation illustrate that model and normalization method selection do affect downstream results, they are only suggestive of theoretical properties of these specific methods that should be further explored. Also, a researcher's primary modeling goal (i.e. predictive accuracy vs. explanatory model) will determine both appropriate model and pre-processing technique selection. In addition, the main goal of normalization is to put features on comparable scale for improved model fitting, performance, and interpretability. Considering normalization as a model selection procedure and selecting based on minimized loss function value (MSE or 0-1 in this case) can potentially lead to overfitting. Finally, this study considers a limited selection of models and model performance measures, while assuming all other proposed aspects of the model development framework are held constant. A more exhaustive study of performance assessments should be considered to better establish the downstream analysis effects of statistical procedures, including coverage probabilities, misclassification rates, sensitivity/specificity, etc. In this study, we selected MSE and 0-1 loss due to the ability to generalize these loss functions across multiple data types and model applications. In addition, these assessments need to include additional consideration on various combinations of model development strategies within the rest of the proposed framework, i.e. sample selection, feature engineering, model validation, etc. The combinations to consider are vast, but considering them within a unified framework allows for a baseline and consistency for continued algorithmic research and applications in the field of data science.

6.1.2 Future Research

To strengthen findings used to propose the usefulness and empirically test the model development framework, it is necessary to explore the theoretical connections between the empirical results and the suggested decision points in the framework. The theoretical work is recommended to justify application of the empirical findings to the proposed framework, creating a more robust theoretical framework for data science. At this stage of the research, the framework and the empirical study results are suggestive of characteristics of linear models, and some aspects of simple non-linear models. Suggested theoretical work will build upon what we know in linear models to extend to non-linear, more complex data structures and models. By building upon foundational knowledge from linear models, we are then completing the theoretical framework in data science using a process much like proof by induction, i.e. if generalized linear models by way of maximum likelihood estimation are, in fact, unbiased in their predictions (base case), and this holds true for any locally linear element of other more complex, globally non-linear models (induction step), then this will be foundational theoretical knowledge for building and testing any models within the proposed model development framework.

6.1.3 Motivating Example

In the empirical study of the model development framework, generalized linear models (GLM) were most often found to have best risk function results regardless of data structure or selected normalization. Maximum likelihood is the best linear unbiased estimator (BLUE) for parameters. If we assume that the maximum likelihood estimate (MLE) is unbiased in GLM as well, then estimates of bias should resolve towards zero as the sample size increases. In this case, the bias-variance decomposition of the loss will be completely

defined by variance. As an extension, the Cramer-Rao lower bound property of MLE (the lower bound of the variance of the estimator) suggests that no other model will achieve a better result of the bias-variance loss decomposition (since bias of zero plus lowest bound of variance equals smallest possible loss, if we assume average error is zero in unbiased model).

This potential explanation requires theoretical understanding of at least two questions: 1) Are generalized linear models, in fact, unbiased? and 2) Does the Cramer-Rao lower bound theorem apply to variance of the prediction and not just the parameters? We can show that, for finite samples, the GLM estimates are biased but they are asymptotically unbiased. If we consider a simple logistic regression with binary dependent variable Y, intercept β_0, and a single binary independent variable X then

$$\Pr(Y_i = 1 \mid X_i = 1) = \Lambda(\beta_0 + \beta X_i) \tag{6.2}$$

where Λ is the logistic link function. The logit form of the simple logistic regression is then

$$\ln\left(\frac{\Pr(Y_i = 1 \mid X_i = 1)}{1 - \Pr(Y_i = 1 \mid X_i = 1)}\right) = \beta_0 + \beta X_i \tag{6.3}$$

If we have a sample size n then n_1 is the number of observations where $X_i = 1$ and n_0 is the number of observations where $X_i = 0$, and $n_1 + n_0 = n$. The estimated conditional probabilities are then

$$\hat{\Pr}(Y = 1 \mid X = 1) \equiv \hat{P}_{1|1} = \frac{1}{n_1} \sum_{X_i=1} y_i \tag{6.4}$$

$$\hat{\Pr}(Y = 1 \mid X = 0) \equiv \hat{P}_{1|0} = \frac{1}{n_0} \sum_{X_i=0} y_i \tag{6.5}$$

and the solutions for the maximum likelihood parameter estimates solve as

$$\hat{\beta_0} = \ln\left(\frac{\hat{P}_{1|0}}{1-\hat{P}_{1|0}}\right), \qquad \hat{\beta} = \ln\left(\frac{\hat{P}_{1|1}}{1-\hat{P}_{1|1}}\right) - \ln\left(\frac{\hat{P}_{1|0}}{1-\hat{P}_{1|0}}\right) \tag{6.6}$$

Due to the properties of MLE for parameter estimates, $\hat{P}_{1|1}$ and $\hat{P}_{1|0}$ are unbiased estimators of the probabilities, but the estimates of $\hat{\beta_0}$ and $\hat{\beta}$ are biased due to the non-linear log transformation. However, since the MLE probability estimates are consistent and asympotically normal, the MLE parameter estimates become asymptotically unbiased as in

$$\lim_{n\to\infty} E[\hat{\beta_0}] = E\left[\ln\left(\lim_{n\to\infty}\frac{\hat{P}_{1|0}}{1-\hat{P}_{1|0}}\right)\right] = E\left[\ln\left(\frac{P_{1|0}}{1-P_{1|0}}\right)\right] = \beta_0 \tag{6.7}$$

and the same holds true for β.

For question 2 of our assumptions, there is no asymptotic closed-form solution for the MLE of the variance-covariance matrix. However, the asymptotic properties of MLE lead to an estimate of the variance-covariance matrix as the inverse of the Hessian matrix of the log-likelihood of the sample

$$Var(\hat{\theta}) \approx -\frac{1}{n}(E[H])^{-1} \approx -\frac{1}{n}\left(\frac{1}{n}\hat{H}\right)^{-1} = -\hat{H}^{-1} \tag{6.8}$$

indicating that the lower bound of the variance of the MLE estimates in a simple logistic regression holds true for large samples, but more work needs to be conducted to extend this to the variance of Y(Agresti, 2003).

6.1.4 Extensions of Research

If the initial empirical study of the model development framework suggests some theoretical basis in linear models, how then do we extend what we know about generalized linear models to other more complex, non-linear models? Once we establish the validity of the foundational assumptions

surrounding the effectiveness of the linear models, then we can slowly increase and test on additional model complexities. For example, Lili Zhang developed a "penalized log-likelihood function" for imbalanced data "by including penalty weights as decision variables for observations in the minority class (i.e. event) and learning them from data along with model coefficients/parameters(Zhang et al., 2019)." These learned weights add one layer of non-linearity to the GLM model and present a logical next step in the theoretical research once we establish that the simple GLM estimates themselves are unbiased.

Another extension of this theoretical research could be to consider the relevance of the GLM findings to a non-probabilistic linear classifier such as linear Support Vector Machines (SVM). In fact, Franc et al. have already established how a linear SVM can be reparameterized as an unbiased maximum likelihood estimate of a probabilistic model(Franc, Zien, and Schölkopf, 2011). However, this work does not yet extend to non-linear SVMs, i.e. those with non-linear kernels replacing the dot product.

Finally, extending the theoretical justification of the model development framework towards neural networks, application of the GLM findings can be tested specifically on extreme learning machines. These neural networks use single or multiple hidden layers where the parameters are randomly assigned, do not need to be tuned, and are not trained using backpropagation. Generally, these networks apply a linear combination of random weights to create a binary output, much like an MLE-type process(Huang, Zhu, and Siew, 2006).

6.1.5 Application to Ethical Principles of Data Science

A unified model development framework lends itself to an improved ethical application of data science. The Analytics and Data Science Institute at Kennesaw State University Principles of Ethical Data Science include:

1. Principle of responsible data collection and sourcing
2. Principle of protection
3. Principle of transparency and reproducibility
4. Principle of foresight
5. Principle of competence

By considering data science research and applications within the context of a unified framework, one can better ensure consistency of research methodologies. For example, if Data Science practitioners followed an agreed upon workflow, this leads to transparency and reproducibility of future work. In the Silberzahn example assessing soccer refereeing (2018), many research teams followed statistically valid yet highly disparate workflows to arrive at a wide range of conclusions. If independent research teams follow validated, unified pathways of research using the same data and assumptions, it is expected that the selection of methods will be more consistent and lead to improved generalization of results. A unified framework also lends itself to the Principle of Competence as it gives data science practitioners a toolbox in which to appropriately use and assess data, much like differential diagnoses used by medical professionals.

Appendix A

Tables

A.0.1 Simulations

Bivariate Normal Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile Transform		Quantile Normalize	
Bivariate Normal - Binary Target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.290	0.290	0.462	0.462	0.474	0.474	0.485	0.485	0.482	0.482	0.290	0.290
		Bias	0.290	0.290	0.220	0.290	0.228	0.290	0.237	0.290	0.234	0.290	0.290	0.290
		Variance	0.000	0.000	0.242	0.409	0.246	0.438	0.438	0.464	0.248	0.457	0.000	0.000
		Noise	0.000	0.000	0.000	0.237	0.000	0.254	0.190	0.268	0.000	0.265	0.000	0.000
		Variance-Bias Ratio	0.000	0.000	1.098	1.412	1.080	1.511	1.850	1.599	1.061	1.578	0.000	0.000
		Percent Change from Raw	~	~	159.319	159.319	163.488	163.488	167.408	167.408	166.279	166.279	100.007	100.007
	Decision Tree													
		Total Loss	0.408	0.408	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.564	0.393	0.393
		Bias	0.250	0.350	0.337	0.710	0.337	0.710	0.337	0.710	0.337	0.710	0.225	0.360
		Variance	0.158	0.232	0.227	0.348	0.227	0.348	0.348	0.348	0.227	0.348	0.168	0.257
		Noise	0.000	0.174	0.000	0.494	0.000	0.494	0.121	0.494	0.000	0.494	0.000	0.223
		Variance-Bias Ratio	0.631	0.663	0.673	0.490	0.673	0.490	1.033	0.490	0.673	0.490	0.748	0.713
		Percent Change from Raw	~	~	138.303	138.303	138.303	138.303	138.303	138.303	138.303	138.303	96.434	96.434
	Random Forest	**Total Loss**	0.295	0.295	0.396	0.396	0.396	0.396	0.396	0.396	0.396	0.396	0.295	0.295
		Bias	0.286	0.290	0.207	0.290	0.207	0.290	0.207	0.290	0.207	0.290	0.286	0.290
		Variance	0.009	0.011	0.188	0.252	0.188	0.252	0.252	0.252	0.188	0.252	0.009	0.011
		Noise	0.000	0.006	0.000	0.146	0.000	0.146	0.064	0.146	0.000	0.146	0.000	0.006
		Variance-Bias Ratio	0.032	0.038	0.909	0.869	0.909	0.869	1.215	0.869	0.909	0.869	0.033	0.038
		Percent Change from Raw	~	~	134.241	134.241	134.241	134.241	134.241	134.241	134.241	134.241	100.002	100.002
	SVM	**Total Loss**	0.292	0.292	0.290	0.290	0.290	0.290	0.290	0.290	0.290	0.290	0.292	0.292
		Bias	0.287	0.290	0.290	0.290	0.290	0.290	0.290	0.290	0.290	0.290	0.288	0.290
		Variance	0.005	0.005	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.005
		Noise	0.000	0.004	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003
		Variance-Bias Ratio	0.016	0.018	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.016	0.018
		Percent Change from Raw	~	~	99.440	99.440	99.440	99.440	99.440	99.440	99.440	99.440	100.218	100.218
	Gradient Boosting	**Total Loss**	0.332	0.332	0.655	0.655	0.655	0.655	0.655	0.655	0.655	0.655	0.332	0.332
		Bias	0.249	0.293	0.540	0.710	0.540	0.710	0.541	0.710	0.540	0.710	0.249	0.297
		Variance	0.082	0.111	0.115	0.132	0.115	0.132	0.132	0.132	0.115	0.132	0.083	0.111
		Noise	0.000	0.073	0.000	0.187	0.000	0.187	0.018	0.187	0.000	0.187	0.000	0.076
		Variance-Bias Ratio	0.331	0.380	0.212	0.186	0.212	0.186	0.244	0.186	0.212	0.186	0.331	0.375
		Percent Change from Raw	~	~	197.402	197.402	197.402	197.402	197.529	197.529	197.402	197.402	100.040	100.040
	Neural Network	**Total Loss**	0.403	0.398	0.427	0.424	0.410	0.411	0.401	0.401	0.401	0.401	0.387	0.381
		Bias	0.206	0.290	0.207	0.290	0.206	0.290	0.207	0.290	0.206	0.290	0.211	0.290
		Variance	0.197	0.258	0.219	0.319	0.204	0.288	0.288	0.288	0.195	0.263	0.176	0.215
		Noise	0.000	0.149	0.000	0.185	0.000	0.167	0.093	0.177	0.000	0.153	0.000	0.124
		Variance-Bias Ratio	0.954	0.888	1.059	1.101	0.993	0.992	1.392	0.992	0.943	0.908	0.835	0.740
		Percent Change from Raw	~	~	105.818	106.440	101.766	103.111	99.415	100.547	99.543	100.548	95.891	95.508

TABLE A.1: Bias-Variance Decomposition Results for Bivariate Normal Data with Binary Target

Bivariate Normal Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Bivariate Normal - Continuous Target	**Linear**	**Type of Loss**	**MSE**	**MSE**	**MSE**	**MSE**	**MSE**	**MSE**
		Total Loss	0.338	200151.804	8999347.482	2146481764.484	2247609.346	0.344
		Bias	0.335	200151.449	8999335.213	2146478729.725	2247069.765	0.338
		Variance	0.003	0.355	12.269	12.269	539.581	0.006
		Noise	0.000	0.000	0.000	3022.490	0.000	0.000
		Variance-Bias Ratio	0.008	0.000	0.000	0.000	0.000	0.019
		Percent Change from Raw	~	59281297.280	2665441848.182	635748573177.009	665700710.123	101.929
	Decision Tree							
		Total Loss	0.748	114.588	114.588	114.588	114.588	0.589
		Bias	0.444	111.856	111.856	111.856	111.856	0.338
		Variance	0.304	2.731	2.731	2.731	2.731	0.251
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.685	0.024	0.024	0.024	0.024	0.742
		Percent Change from Raw	~	15311.228	15311.228	15311.228	15311.228	78.694
	Random Forest	**Total Loss**	2.809	22.970	22.970	22.970	22.970	2.812
		Bias	2.425	22.501	22.501	22.501	22.501	2.427
		Variance	0.385	0.469	0.469	0.469	0.469	0.385
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.159	0.021	0.021	0.021	0.021	0.158
		Percent Change from Raw	~	817.686	817.686	817.686	817.686	100.088
	SVM	**Total Loss**	0.652	8.712	8.884	8.840	8.878	0.662
		Bias	0.623	8.632	8.771	8.747	8.768	0.629
		Variance	0.029	0.080	0.112	0.112	0.110	0.033
		Noise	0.000	0.000	0.000	0.019	0.000	0.000
		Variance-Bias Ratio	0.047	0.009	0.013	0.013	0.013	0.052
		Percent Change from Raw	~	1336.239	1362.522	1355.909	1361.662	101.476
	Gradient Boosting	**Total Loss**	0.550	151.982	151.982	152.078	151.982	0.552
		Bias	0.289	150.938	150.938	151.034	150.938	0.290
		Variance	0.261	1.043	1.043	1.043	1.043	0.262
		Noise	0.000	0.000	0.000	0.001	0.000	0.000
		Variance-Bias Ratio	0.904	0.007	0.007	0.007	0.007	0.901
		Percent Change from Raw	~	27615.537	27615.537	27633.076	27615.537	100.363
	Neural Network	**Total Loss**	0.341	200151.851	8999382.281	2145420994.196	2247611.004	0.356
		Bias	0.335	200151.497	8999370.017	2145297375.914	2247071.513	0.315
		Variance	0.007	0.355	12.264	12.264	539.487	0.041
		Noise	0.000	0.000	0.000	123606.017	0.000	0.000
		Variance-Bias Ratio	0.020	0.000	0.000	0.000	0.000	0.132
		Percent Change from Raw	~	58619129.637	2635678625.092	628336487970.484	658265211.621	104.238

TABLE A.2: Bias-Variance Decomposition Results for Bivariate Normal Data with Continuous Target

Bivariate Normal Data with Poisson Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Bivariate Normal - Poisson Target	**Poisson Regression**	**Type of Loss**	**MSE**	**MSE**	**MSE**	**MSE**	**MSE**	**MSE**
		Total Loss	1.502	191.901	1.874	1.764	3.004	1.502
		Bias	1.280	1.561	1.609	1.600	1.459	1.275
		Variance	0.222	190.340	0.264	0.264	1.546	0.227
		Noise	0.000	0.000	0.000	0.101	0.000	0.000
		Variance-Bias Ratio	0.173	121.919	0.164	0.165	1.060	0.178
		Percent Change from Raw	~	12776.852	124.753	117.429	200.027	100.023
	Decision Tree							
		Total Loss	1.869	1.801	1.801	1.802	1.801	2.065
		Bias	1.126	1.428	1.428	1.429	1.428	1.173
		Variance	0.743	0.373	0.373	0.373	0.373	0.892
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.660	0.261	0.261	0.261	0.261	0.760
		Percent Change from Raw	~	96.319	96.319	96.371	96.319	110.450
	Random Forest	**Total Loss**	1.564	1.125	1.125	1.125	1.125	1.564
		Bias	1.369	0.980	0.980	0.980	0.980	1.370
		Variance	0.195	0.145	0.145	0.145	0.145	0.194
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.142	0.148	0.148	0.148	0.148	0.142
		Percent Change from Raw	~	71.964	71.964	71.964	71.964	100.059
	SVM	**Total Loss**	1.676	1.324	1.443	1.886	1.341	1.700
		Bias	1.528	1.091	1.192	1.805	1.104	1.556
		Variance	0.148	0.233	0.252	0.252	0.237	0.143
		Noise	0.000	0.000	0.000	0.171	0.000	0.000
		Variance-Bias Ratio	0.097	0.213	0.211	0.139	0.214	0.092
		Percent Change from Raw	~	78.974	86.122	112.515	80.023	101.403
	Gradient Boosting	**Total Loss**	1.561	1.167	1.167	1.167	1.167	1.550
		Bias	1.325	0.943	0.943	0.943	0.943	1.319
		Variance	0.235	0.224	0.224	0.224	0.224	0.231
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.178	0.237	0.237	0.237	0.237	0.175
		Percent Change from Raw	~	74.795	74.795	74.791	74.795	99.310
	Neural Network	**Total Loss**	1.501	50.236	2611.864	553224.671	740.072	1.496
		Bias	1.253	25.275	1286.970	257374.377	412.057	1.267
		Variance	0.248	24.961	1324.894	1324.894	328.015	0.229
		Noise	0.000	0.000	0.000	294525.400	0.000	0.000
		Variance-Bias Ratio	0.198	0.988	1.029	0.005	0.796	0.181
		Percent Change from Raw	~	3346.523	173990.881	36853390.488	49300.337	99.642

TABLE A.3: Bias-Variance Decomposition Results for Bivariate Normal Data with Poisson Target

Ranked Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile		Upper Quartile	
Ranked - Binary Target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.444	0.444	0.505	0.505	0.506	0.506	0.506	0.506	0.506	0.506	0.444	0.444
		Bias	0.369	0.437	0.343	0.500	0.344	0.507	0.344	0.510	0.343	0.503	0.369	0.437
		Variance	0.075	0.111	0.163	0.240	0.162	0.240	0.240	0.240	0.163	0.241	0.075	0.111
		Noise	0.000	0.104	0.000	0.235	0.000	0.241	0.078	0.244	0.000	0.238	0.000	0.104
		Variance-Bias Ratio	0.202	0.255	0.474	0.480	0.473	0.474	0.698	0.471	0.476	0.479	0.202	0.255
		Percent Change from Raw	~	~	113.941	113.941	114.128	114.128	114.127	114.127	114.080	114.080	100.000	100.000
	Decision Tree													
		Total Loss	0.489	0.489	0.494	0.494	0.494	0.494	0.494	0.494	0.494	0.494	0.493	0.493
		Bias	0.292	0.437	0.246	0.433	0.246	0.437	0.246	0.437	0.246	0.437	0.291	0.467
		Variance	0.197	0.302	0.248	0.454	0.248	0.454	0.454	0.454	0.248	0.454	0.202	0.314
		Noise	0.000	0.250	0.000	0.393	0.000	0.396	0.206	0.396	0.000	0.396	0.000	0.287
		Variance-Bias Ratio	0.672	0.692	1.009	1.047	1.006	1.040	1.843	1.040	1.006	1.040	0.692	0.672
		Percent Change from Raw	~	~	100.905	100.905	101.041	101.041	101.041	101.041	101.041	101.041	100.767	100.767
	Random Forest	**Total Loss**	0.445	0.445	0.488	0.488	0.487	0.487	0.487	0.487	0.487	0.487	0.445	0.445
		Bias	0.354	0.433	0.248	0.437	0.247	0.437	0.247	0.437	0.247	0.437	0.354	0.433
		Variance	0.091	0.121	0.240	0.398	0.240	0.399	0.399	0.399	0.240	0.399	0.091	0.121
		Noise	0.000	0.109	0.000	0.347	0.000	0.348	0.159	0.348	0.000	0.348	0.000	0.109
		Variance-Bias Ratio	0.257	0.279	0.965	0.912	0.969	0.914	1.613	0.914	0.969	0.914	0.257	0.279
		Percent Change from Raw	~	~	109.658	109.658	109.524	109.524	109.524	109.524	109.524	109.524	100.000	100.000
	SVM	**Total Loss**	0.437	0.437	0.441	0.441	0.460	0.460	0.460	0.460	0.459	0.459	0.437	0.437
		Bias	0.437	0.437	0.407	0.437	0.308	0.437	0.309	0.437	0.313	0.437	0.437	0.437
		Variance	0.000	0.000	0.034	0.035	0.152	0.187	0.187	0.187	0.146	0.178	0.000	0.000
		Noise	0.000	0.000	0.000	0.031	0.000	0.163	0.036	0.163	0.000	0.155	0.000	0.000
		Variance-Bias Ratio	0.000	0.000	0.083	0.080	0.493	0.428	0.606	0.428	0.468	0.408	0.000	0.000
		Percent Change from Raw	~	~	101.023	101.023	105.432	105.432	105.403	105.403	105.171	105.171	100.000	100.000
	Gradient Boosting	**Total Loss**	0.465	0.465	0.521	0.521	0.521	0.521	0.521	0.521	0.521	0.521	0.465	0.465
		Bias	0.303	0.473	0.298	0.563	0.299	0.563	0.299	0.563	0.299	0.563	0.303	0.473
		Variance	0.162	0.241	0.222	0.334	0.222	0.334	0.334	0.334	0.222	0.334	0.162	0.241
		Noise	0.000	0.250	0.000	0.376	0.000	0.376	0.112	0.376	0.000	0.376	0.000	0.250
		Variance-Bias Ratio	0.535	0.509	0.745	0.592	0.745	0.593	1.119	0.593	0.745	0.593	0.535	0.509
		Percent Change from Raw	~	~	112.059	112.059	112.123	112.123	112.123	112.123	112.123	112.123	100.000	100.000
	Neural Network	**Total Loss**	0.478	0.475	0.492	0.492	0.487	0.485	0.486	0.488	0.484	0.486	0.476	0.480
		Bias	0.253	0.437	0.248	0.440	0.249	0.437	0.247	0.437	0.250	0.437	0.258	0.437
		Variance	0.225	0.325	0.245	0.429	0.238	0.383	0.383	0.383	0.234	0.390	0.218	0.353
		Noise	0.000	0.286	0.000	0.377	0.000	0.334	0.144	0.331	0.000	0.341	0.000	0.309
		Variance-Bias Ratio	0.889	0.744	0.988	0.975	0.955	0.876	1.547	0.876	0.937	0.893	0.844	0.808
		Percent Change from Raw	~	~	103.006	103.458	101.848	102.056	101.619	102.607	101.338	102.220	99.489	101.060

TABLE A.4: Bias-Variance Decomposition Results for Ranked Data with Binary Target

Ranked Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Ranked - Continuous Target	Linear	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	0.174	304375397009.301	36278206201 36.140	3635081968755.210	3635123035407.580	0.174
		Bias	0.174	304375397000.036	36278206200 26.510	3635081968645.360	3635121515273.040	0.174
		Variance	0.000	9.265	109.624	109.624	1520134.541	0.000
		Noise	0.000	0.000	0.006	0.226	0.001	0.000
		Variance-Bias Ratio	0.000	0.000	0.000	0.000	0.000	0.000
		Percent Change from Raw	~	175139389533788.000	208746927311 3810.000	209164749571 9290.000	209167112571 2040.000	100.000
	Decision Tree							
		Total Loss	6464.023	3058180.695	3058180.695	3058180.695	3058180.695	6421.793
		Bias	959.928	3055675.410	3055675.410	3055675.410	3055675.410	1113.417
		Variance	5504.094	2505.285	2505.285	2505.285	2505.285	5308.376
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	5.734	0.001	0.001	0.001	0.001	4.768
		Percent Change from Raw	~	47310.799	47310.799	47310.799	47310.799	99.347
	Random Forest	**Total Loss**	102252.318	1283608.530	1283608.530	1283608.530	1283608.530	102252.318
		Bias	84950.814	1277680.008	1277680.008	1277680.008	1277680.008	84950.814
		Variance	17301.504	5928.522	5928.522	5928.522	5928.522	17301.504
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.204	0.005	0.005	0.005	0.005	0.204
		Percent Change from Raw	~	1255.334	1255.334	1255.334	1255.334	100.000
	SVM	**Total Loss**	468652.034	468824.111	468659.254	468659.354	468687.960	468652.034
		Bias	467902.232	468167.990	467974.616	467974.484	467999.280	467902.232
		Variance	749.803	656.121	684.638	684.638	688.680	749.803
		Noise	0.000	0.000	0.000	0.232	0.000	0.000
		Variance-Bias Ratio	0.002	0.001	0.001	0.001	0.001	0.002
		Percent Change from Raw	~	100.037	100.002	100.002	100.008	100.000
	Gradient Boosting	**Total Loss**	10562.614	3087157.051	3087157.051	3087157.051	3087157.051	10562.614
		Bias	3197.090	3083591.900	3083591.900	3083591.900	3083591.900	3197.090
		Variance	7365.524	3565.151	3565.151	3565.151	3565.151	7365.524
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	2.304	0.001	0.001	0.001	0.001	2.304
		Percent Change from Raw	~	29227.208	29227.208	29227.208	29227.208	100.000
	Neural Network	**Total Loss**	0.175	304375382136.401	3627820444148.760	3635081654196.280	3635123003365.700	0.175
		Bias	0.174	304375382127.117	3627820444038.680	3635081654084.160	3635121483216.140	0.174
		Variance	0.001	9.284	110.079	110.079	1520149.553	0.001
		Noise	0.000	0.000	0.000	2.041	0.005	0.000
		Variance-Bias Ratio	0.007	0.000	0.000	0.000	0.000	0.007
		Percent Change from Raw	~	173769104867507.000	207113698478 1720.000	207528243820 6230.000	207530604460 9960.000	99.994

TABLE A.5: Bias-Variance Decomposition Results for Ranked Data with Continuous Target

Ranked Data with Poisson Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Ranked - Poisson Target	Poisson Regression	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	1.280	2.000E+18	8.871E+17	8.850E+17	9.325E+17	1.280
		Bias	1.280	1.009E+18	2.303E+17	2.294E+17	2.537E+17	1.280
		Variance	0.000	9.907E+17	6.568E+17	6.568E+17	6.788E+17	0.000
		Noise	0.000	4.096E+03	0.000E+00	1.261E+15	1.024E+03	0.000
		Variance-Bias Ratio	0.000	9.816E-01	2.852E+00	2.863E+00	2.675E+00	0.000
		Percent Change from Raw	~	1.562E+20	6.929E+19	6.913E+19	7.284E+19	100.000
	Decision Tree							
		Total Loss	2.418	2.276	2.270	2.270	2.270	2.305
		Bias	1.318	1.325	1.320	1.320	1.320	1.320
		Variance	1.100	0.950	0.950	0.950	0.950	0.985
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.834	0.717	0.719	0.719	0.719	0.746
		Percent Change from Raw	~	94.094	93.868	93.868	93.868	95.304
	Random Forest	**Total Loss**	1.505	1.307	1.307	1.307	1.307	1.505
		Bias	1.396	1.279	1.279	1.279	1.279	1.396
		Variance	0.108	0.028	0.028	0.028	0.028	0.108
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.077	0.022	0.022	0.022	0.022	0.077
		Percent Change from Raw	~	86.878	86.880	86.880	86.880	100.000
	SVM	**Total Loss**	1.281	1.434	1.553	1.553	1.559	1.281
		Bias	1.280	1.339	1.398	1.398	1.401	1.280
		Variance	0.001	0.096	0.155	0.155	0.159	0.001
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.001	0.071	0.111	0.111	0.113	0.001
		Percent Change from Raw	~	111.913	121.171	121.171	121.664	100.000
	Gradient Boosting	**Total Loss**	1.804	1.999	2.000	2.000	2.000	1.804
		Bias	1.594	1.330	1.330	1.330	1.330	1.594
		Variance	0.210	0.670	0.670	0.670	0.670	0.210
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.132	0.504	0.504	0.504	0.504	0.132
		Percent Change from Raw	~	110.831	110.859	110.855	110.855	100.000
	Neural Network	**Total Loss**	1.345E+00	3.224E+03	3.912E+04	3.922E+04	4.046E+04	1.346E+00
		Bias	1.302E+00	2.120E+03	2.587E+04	2.594E+04	2.718E+04	1.302E+00
		Variance	4.318E-02	1.105E+03	1.325E+04	1.325E+04	1.327E+04	4.328E-02
		Noise	6.661E-15	0.000E+00	1.019E-10	3.600E+01	0.000E+00	0.000E+00
		Variance-Bias Ratio	3.317E-02	5.212E-01	5.121E-01	5.107E-01	4.882E-01	3.323E-02
		Percent Change from Raw	~	2.397E+05	2.908E+06	2.915E+06	3.007E+06	1.000E+02

TABLE A.6: Bias-Variance Decomposition Results for Ranked Data with Poisson Target

Categorical Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile		Upper Quartile	
Categorical - Binary Target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.473	0.473	0.476	0.476	0.473	0.473	0.473	0.473	0.473	0.473	0.473	0.473
		Bias	0.281	0.447	0.266	0.447	0.281	0.447	0.281	0.447	0.281	0.447	0.281	0.447
		Variance	0.192	0.282	0.209	0.322	0.192	0.282	0.282	0.282	0.192	0.282	0.192	0.282
		Noise	0.000	0.256	0.000	0.293	0.000	0.256	0.090	0.256	0.000	0.256	0.000	0.256
		Variance-Bias Ratio	0.683	0.632	0.786	0.721	0.683	0.632	1.004	0.632	0.683	0.632	0.683	0.632
		Percent Change from Raw	~	~	100.573	100.573	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Decision Tree													
		Total Loss	0.502	0.502	0.480	0.480	0.485	0.485	0.485	0.485	0.485	0.485	0.485	0.485
		Bias	0.338	0.550	0.281	0.487	0.306	0.497	0.306	0.497	0.306	0.497	0.306	0.497
		Variance	0.164	0.264	0.200	0.320	0.178	0.284	0.284	0.284	0.178	0.284	0.178	0.284
		Noise	0.000	0.311	0.000	0.326	0.000	0.295	0.105	0.295	0.000	0.295	0.000	0.295
		Variance-Bias Ratio	0.484	0.479	0.710	0.657	0.582	0.571	0.926	0.571	0.582	0.571	0.582	0.571
		Percent Change from Raw	~	~	95.636	95.636	96.508	96.508	96.508	96.508	96.508	96.508	96.508	96.508
	Random Forest	**Total Loss**	0.476	0.476	0.484	0.484	0.476	0.476	0.476	0.476	0.476	0.476	0.476	0.476
		Bias	0.288	0.450	0.282	0.487	0.288	0.450	0.288	0.450	0.288	0.450	0.288	0.450
		Variance	0.187	0.272	0.201	0.311	0.187	0.272	0.272	0.272	0.187	0.272	0.187	0.272
		Noise	0.000	0.247	0.000	0.314	0.000	0.247	0.085	0.247	0.000	0.247	0.000	0.247
		Variance-Bias Ratio	0.649	0.605	0.714	0.638	0.649	0.605	0.944	0.605	0.649	0.605	0.649	0.605
		Percent Change from Raw	~	~	101.709	101.709	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	SVM	**Total Loss**	0.476	0.476	0.473	0.473	0.476	0.476	0.476	0.476	0.476	0.476	0.476	0.476
		Bias	0.288	0.447	0.285	0.437	0.288	0.447	0.288	0.447	0.288	0.447	0.288	0.447
		Variance	0.188	0.275	0.187	0.285	0.188	0.275	0.275	0.275	0.188	0.275	0.188	0.275
		Noise	0.000	0.246	0.000	0.249	0.000	0.246	0.087	0.246	0.000	0.246	0.000	0.246
		Variance-Bias Ratio	0.653	0.616	0.656	0.652	0.653	0.616	0.957	0.616	0.653	0.616	0.653	0.616
		Percent Change from Raw	~	~	99.398	99.398	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Gradient Boosting	**Total Loss**	0.483	0.483	0.496	0.496	0.483	0.483	0.483	0.483	0.483	0.483	0.483	0.483
		Bias	0.305	0.447	0.326	0.500	0.305	0.447	0.305	0.447	0.305	0.447	0.305	0.447
		Variance	0.178	0.284	0.170	0.234	0.178	0.284	0.284	0.284	0.178	0.284	0.178	0.284
		Noise	0.000	0.248	0.000	0.238	0.000	0.248	0.106	0.248	0.000	0.248	0.000	0.248
		Variance-Bias Ratio	0.585	0.635	0.521	0.469	0.585	0.635	0.932	0.635	0.585	0.635	0.585	0.635
		Percent Change from Raw	~	~	102.711	102.711	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Neural Network	**Total Loss**	0.491	0.490	0.496	0.496	0.491	0.490	0.491	0.490	0.491	0.491	0.490	0.490
		Bias	0.248	0.447	0.248	0.447	0.249	0.447	0.249	0.447	0.249	0.447	0.249	0.450
		Variance	0.242	0.419	0.249	0.468	0.242	0.418	0.418	0.418	0.242	0.422	0.241	0.418
		Noise	0.000	0.376	0.000	0.419	0.000	0.375	0.177	0.375	0.000	0.377	0.000	0.378
		Variance-Bias Ratio	0.975	0.939	1.005	1.048	0.973	0.936	1.676	0.936	0.974	0.944	0.970	0.928
		Percent Change from Raw	~	~	101.214	101.205	100.004	99.975	100.041	100.001	100.097	100.151	99.941	99.996

TABLE A.7: Bias-Variance Decomposition Results for Categorical Data with Binary Target

Categorical Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Categorical - Continuous Target	Linear	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	0.243	341498778613906000.000	0.243	0.243	12972148828.537	0.243
		Bias	0.241	251811327306207000.000	0.241	0.241	27556466.022	0.241
		Variance	0.003	89687451307698700.000	0.003	0.003	12944592362.515	0.003
		Noise	0.000	3584.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.011	0.356	0.011	0.011	469.748	0.011
		Percent Change from Raw	~	14033812968227100000000.000	100.000	100.000	5330874423463.480	100.020
	Decision Tree							
		Total Loss	0.245	1.325	0.243	0.243	0.243	0.243
		Bias	0.239	1.198	0.240	0.240	0.240	0.240
		Variance	0.006	0.126	0.003	0.003	0.003	0.003
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.025	0.105	0.013	0.013	0.013	0.013
		Percent Change from Raw	~	540.783	99.133	99.133	99.133	99.133
	Random Forest	**Total Loss**	0.364	1.605	0.364	0.364	0.364	0.364
		Bias	0.350	1.489	0.350	0.350	0.350	0.350
		Variance	0.013	0.116	0.013	0.013	0.013	0.013
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.038	0.078	0.038	0.038	0.038	0.038
		Percent Change from Raw	~	441.086	100.000	100.000	100.000	100.000
	SVM	**Total Loss**	0.242	0.689	0.242	0.242	0.242	0.242
		Bias	0.238	0.689	0.238	0.238	0.238	0.238
		Variance	0.004	0.000	0.004	0.004	0.004	0.004
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.016	0.000	0.016	0.016	0.016	0.017
		Percent Change from Raw	~	284.707	100.000	100.000	100.000	99.918
	Gradient Boosting	**Total Loss**	0.243	2.096	0.243	0.243	0.243	0.243
		Bias	0.240	2.096	0.240	0.240	0.240	0.240
		Variance	0.003	0.000	0.003	0.003	0.003	0.003
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.012	0.000	0.012	0.012	0.012	0.012
		Percent Change from Raw	~	862.207	100.000	100.000	100.000	100.000
	Neural Network	**Total Loss**	0.244	0.792	0.244	0.244	0.244	0.244
		Bias	0.241	0.623	0.241	0.241	0.241	0.241
		Variance	0.002	0.169	0.002	0.002	0.002	0.002
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.010	0.271	0.010	0.010	0.010	0.010
		Percent Change from Raw	~	325.239	100.000	100.041	100.020	99.980

TABLE A.8: Bias-Variance Decomposition Results for Categorical Data with Continuous Target

Categorical Data with Poisson Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Categorical - Poisson Target	Poisson Regression	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	1.753	1.748	1.753	1.753	1.753	1.753
		Bias	1.682	1.673	1.682	1.682	1.682	1.682
		Variance	0.071	0.074	0.071	0.071	0.071	0.071
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.042	0.044	0.042	0.042	0.042	0.042
		Percent Change from Raw	~	99.698	100.000	100.000	100.000	100.000
	Decision Tree							
		Total Loss	1.559	1.591	1.499	1.499	1.499	1.499
		Bias	1.351	1.435	1.314	1.314	1.314	1.314
		Variance	0.208	0.157	0.185	0.185	0.185	0.185
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.154	0.109	0.141	0.141	0.141	0.141
		Percent Change from Raw	~	102.089	96.164	96.164	96.164	96.164
	Random Forest	**Total Loss**	1.581	1.661	1.581	1.581	1.581	1.581
		Bias	1.434	1.559	1.434	1.434	1.434	1.434
		Variance	0.147	0.102	0.147	0.147	0.147	0.147
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.103	0.066	0.103	0.103	0.103	0.103
		Percent Change from Raw	~	105.064	100.000	100.000	100.000	100.000
	SVM	**Total Loss**	1.754	1.783	1.754	1.754	1.754	1.754
		Bias	1.685	1.740	1.685	1.685	1.685	1.685
		Variance	0.069	0.043	0.069	0.069	0.069	0.069
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.041	0.025	0.041	0.041	0.041	0.041
		Percent Change from Raw	~	101.675	100.000	100.000	100.000	100.000
	Gradient Boosting	**Total Loss**	1.521	1.544	1.521	1.521	1.521	1.521
		Bias	1.343	1.378	1.343	1.343	1.343	1.343
		Variance	0.178	0.166	0.178	0.178	0.178	0.178
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.133	0.121	0.133	0.133	0.133	0.133
		Percent Change from Raw	~	101.547	100.000	100.000	100.000	100.000
	Neural Network	**Total Loss**	1.554	1.525	1.554	1.554	1.553	1.554
		Bias	1.405	1.301	1.405	1.405	1.404	1.405
		Variance	0.149	0.224	0.149	0.149	0.149	0.149
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.106	0.172	0.106	0.106	0.106	0.106
		Percent Change from Raw	~	98.134	100.007	99.993	99.947	99.980

TABLE A.9: Bias-Variance Decomposition Results for Categorical Data with Poisson Target

Mixed Data with Binary Target

Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile		Upper Quartile	
Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
	Total Loss	0.499	0.499	0.499	0.499	0.499	0.499	0.497	0.497	0.499	0.499	0.496	0.496
	Bias	0.311	0.510	0.250	0.480	0.250	0.480	0.253	0.480	0.250	0.480	0.303	0.477
	Variance	0.188	0.286	0.249	0.474	0.250	0.479	0.479	0.421	0.249	0.471	0.193	0.297
	Noise	0.000	0.297	0.000	0.455	0.000	0.460	0.235	0.404	0.000	0.452	0.000	0.278
	Variance-Bias Ratio	0.605	0.561	0.999	0.989	1.000	0.997	1.891	0.877	0.998	0.982	0.638	0.623
	Percent Change from Raw	~	~	99.962	99.962	99.988	99.988	99.521	99.521	99.931	99.931	99.254	99.254
Decision Tree													
	Total Loss	0.479	0.479	0.504	0.504	0.503	0.503	0.503	0.503	0.503	0.503	0.492	0.492
	Bias	0.267	0.473	0.268	0.520	0.267	0.520	0.267	0.520	0.267	0.520	0.287	0.470
	Variance	0.212	0.336	0.236	0.387	0.236	0.389	0.389	0.389	0.236	0.389	0.204	0.317
	Noise	0.000	0.330	0.000	0.403	0.000	0.405	0.153	0.405	0.000	0.405	0.000	0.296
	Variance-Bias Ratio	0.793	0.709	0.879	0.744	0.884	0.747	1.455	0.747	0.884	0.747	0.710	0.675
	Percent Change from Raw	~	~	105.113	105.113	104.978	104.978	104.978	104.978	104.978	104.978	102.570	102.570
Random Forest	**Total Loss**	0.503	0.503	0.487	0.487	0.490	0.490	0.490	0.490	0.490	0.490	0.503	0.503
	Bias	0.294	0.510	0.357	0.480	0.353	0.480	0.353	0.480	0.353	0.480	0.295	0.510
	Variance	0.209	0.329	0.129	0.153	0.136	0.165	0.165	0.165	0.136	0.165	0.209	0.329
	Noise	0.000	0.336	0.000	0.146	0.000	0.155	0.028	0.155	0.000	0.155	0.000	0.336
	Variance-Bias Ratio	0.709	0.646	0.362	0.319	0.386	0.343	0.466	0.343	0.386	0.343	0.709	0.646
	Percent Change from Raw	~	~	96.736	96.736	97.287	97.287	97.287	97.287	97.287	97.287	100.023	100.023
SVM	**Total Loss**	0.504	0.504	0.484	0.484	0.484	0.484	0.489	0.489	0.482	0.482	0.505	0.505
	Bias	0.309	0.533	0.387	0.480	0.396	0.480	0.316	0.480	0.441	0.480	0.311	0.520
	Variance	0.194	0.299	0.097	0.109	0.088	0.097	0.097	0.097	0.041	0.043	0.194	0.299
	Noise	0.000	0.328	0.000	0.105	0.000	0.093	0.076	0.088	0.000	0.041	0.000	0.314
	Variance-Bias Ratio	0.628	0.560	0.251	0.227	0.221	0.202	0.307	0.202	0.093	0.090	0.624	0.575
	Percent Change from Raw	~	~	96.131	96.131	96.036	96.036	97.036	97.036	95.607	95.607	100.269	100.269
Gradient Boosting	**Total Loss**	0.507	0.507	0.503	0.503	0.505	0.505	0.505	0.505	0.505	0.505	0.507	0.507
	Bias	0.318	0.523	0.257	0.537	0.260	0.523	0.260	0.523	0.260	0.523	0.318	0.523
	Variance	0.189	0.286	0.246	0.447	0.245	0.437	0.437	0.437	0.245	0.437	0.189	0.286
	Noise	0.000	0.303	0.000	0.481	0.000	0.456	0.192	0.456	0.000	0.456	0.000	0.302
	Variance-Bias Ratio	0.595	0.547	0.960	0.833	0.940	0.835	1.680	0.835	0.940	0.835	0.594	0.546
	Percent Change from Raw	~	~	99.241	99.241	99.599	99.599	99.599	99.599	99.599	99.599	100.082	100.082
Neural Network	**Total Loss**	0.498	0.498	0.498	0.499	0.499	0.499	0.499	0.499	0.499	0.498	0.497	0.497
	Bias	0.250	0.480	0.250	0.480	0.250	0.480	0.250	0.480	0.250	0.480	0.251	0.480
	Variance	0.248	0.437	0.248	0.462	0.249	0.478	0.478	0.478	0.250	0.462	0.246	0.439
	Noise	0.000	0.419	0.000	0.443	0.000	0.459	0.228	0.459	0.000	0.444	0.000	0.422
	Variance-Bias Ratio	0.992	0.910	0.993	0.962	0.998	0.996	1.915	0.996	1.000	0.963	0.982	0.915
	Percent Change from Raw	~	~	99.986	100.167	100.084	100.286	100.183	100.286	100.181	100.149	99.700	99.912

TABLE A.10: Bias-Variance Decomposition Results for Mixed Data with Binary Target

Mixed Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Mixed Data - Continuous Target	Linear	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	0.367	308689815305532000 0.000	5568986.309	2121588770.884	1895990182.853	0.363
		Bias	0.362	206007019532614000 0.000	5568970.874	2121584456.477	886.312	0.355
		Variance	0.005	102682795772917000 0.000	15.435	15.435	1895989296.541	0.008
		Noise	0.000	10240.000	0.000	4298.972	0.000	0.000
		Variance-Bias Ratio	0.014	0.498	0.000	0.000	2139189.427	0.022
		Percent Change from Raw	~	841116717750002000000.000	1517435060.458	578089621005.122	516618612087.659	98.992
	Decision Tree							
		Total Loss	0.955	74.278	74.222	74.222	74.222	1.193
		Bias	0.472	73.880	73.821	73.817	73.821	0.717
		Variance	0.483	0.398	0.401	0.401	0.401	0.476
		Noise	0.000	0.000	0.000	0.004	0.000	0.000
		Bias-Variance Ratio	1.023	0.005	0.005	0.005	0.005	0.664
		Percent Change from Raw	~	7777.318	7771.442	7771.406	7771.442	124.898
	Random Forest	**Total Loss**	3.925	23.205	23.205	23.205	23.205	3.921
		Bias	3.506	22.923	22.923	22.923	22.923	3.502
		Variance	0.419	0.282	0.282	0.282	0.282	0.419
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Bias-Variance Ratio	0.120	0.012	0.012	0.012	0.012	0.120
		Percent Change from Raw	~	591.191	591.191	591.191	591.191	99.893
	SVM	**Total Loss**	0.816	11.521	11.548	11.528	12.201	0.831
		Bias	0.783	11.519	11.528	11.521	11.952	0.801
		Variance	0.032	0.002	0.021	0.021	0.250	0.031
		Noise	0.000	0.000	0.000	0.014	0.000	0.000
		Bias-Variance Ratio	0.041	0.000	0.002	0.002	0.021	0.038
		Percent Change from Raw	~	1412.698	1416.026	1413.574	1496.065	101.955
	Gradient Boosting	**Total Loss**	1.695	59.055	62.797	62.806	62.797	1.694
		Bias	1.154	58.342	62.004	62.013	62.004	1.151
		Variance	0.541	0.712	0.792	0.792	0.792	0.543
		Noise	0.000	0.000	0.000	0.001	0.000	0.000
		Bias-Variance Ratio	0.469	0.012	0.013	0.013	0.013	0.472
		Percent Change from Raw	~	3483.593	3704.328	3704.895	3704.328	99.940
	Neural Network	**Total Loss**	0.366	172936.383	5567736.822	2092341169.038	1980462.195	0.425
		Bias	0.359	172879.121	5567709.809	2077346984.312	1980266.673	0.355
		Variance	0.007	57.262	27.013	27.013	195.522	0.070
		Noise	0.000	0.000	0.000	14994157.713	0.000	0.000
		Bias-Variance Ratio	0.020	0.000	0.000	0.000	0.000	0.198
		Percent Change from Raw	~	47277162.892	1522101918.052	572001983668.337	541416629.719	116.261

TABLE A.11: Bias-Variance Decomposition Results for Mixed Data with Continuous Target

Mixed Data with Poisson Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile	Upper Quartile
Categorical - Continuous Target	Poisson Regression	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	1.929	631.554	2.208	2.070	6.928	1.929
		Bias	1.743	6.792	1.801	1.928	1.271	1.740
		Variance	0.186	624.762	0.407	0.407	5.657	0.188
		Noise	0.000	0.000	0.000	0.264	0.000	0.000
		Variance-Bias Ratio	0.107	91.989	0.226	0.211	4.452	0.108
		Percent Change from Raw	~	32734.401	114.423	107.289	359.076	99.957
	Decision Tree							
		Total Loss	2.087	2.127	2.137	2.140	2.137	2.119
		Bias	1.123	1.199	1.213	1.212	1.213	1.264
		Variance	0.964	0.927	0.924	0.924	0.924	0.855
		Noise	0.000	0.000	0.000	0.004	0.000	0.000
		Variance-Bias Ratio	0.859	0.773	0.762	0.762	0.762	0.677
		Percent Change from Raw	~	101.899	102.382	102.521	102.382	101.557
	Random Forest	**Total Loss**	1.820	2.129	2.113	2.113	2.113	1.821
		Bias	1.608	2.029	1.997	1.997	1.997	1.609
		Variance	0.212	0.100	0.116	0.116	0.116	0.212
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.132	0.049	0.058	0.058	0.058	0.132
		Percent Change from Raw	~	116.956	116.100	116.100	116.100	100.040
	SVM	**Total Loss**	1.940	2.029	2.201	2.247	2.188	1.938
		Bias	1.775	1.865	2.157	2.244	2.133	1.771
		Variance	0.165	0.164	0.044	0.044	0.055	0.166
		Noise	0.000	0.000	0.000	0.041	0.000	0.000
		Variance-Bias Ratio	0.093	0.088	0.020	0.020	0.026	0.094
		Percent Change from Raw	~	104.577	113.429	115.807	112.765	99.867
	Gradient Boosting	**Total Loss**	1.746	2.042	2.000	2.001	2.000	1.746
		Bias	1.531	1.876	1.787	1.787	1.787	1.530
		Variance	0.216	0.166	0.213	0.213	0.213	0.216
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.141	0.089	0.119	0.119	0.119	0.141
		Percent Change from Raw	~	116.930	114.528	114.553	114.528	99.966
	Neural Network	**Total Loss**	1.892	50.960	2168.995	531230.373	508.503	1.882
		Bias	1.711	23.142	938.748	185620.790	157.692	1.708
		Variance	0.181	27.817	1230.247	1230.247	350.811	0.175
		Noise	0.000	0.000	0.000	344379.336	0.000	0.000
		Variance-Bias Ratio	0.106	1.202	1.311	0.007	2.225	0.102
		Percent Change from Raw	~	2693.431	114640.142	28077665.821	26876.443	99.497

TABLE A.12: Bias-Variance Decomposition Results for Mixed Data with Poisson Target

A.0.2 Benchmark Results

Wine Quality Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile Transform		Quantile Normalize	
Wine Quality with binary target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.038	0.038	0.058	0.058	0.100	0.100	0.166	0.166	0.062	0.062	0.038	0.038
		Bias	0.037	0.038	0.036	0.037	0.037	0.037	0.047	0.037	0.034	0.037	0.037	0.037
		Variance	0.000	0.001	0.022	0.023	0.063	0.068	0.144	0.140	0.028	0.030	0.001	0.001
		Noise	0.000	0.001	0.000	0.002	0.000	0.005	0.024	0.011	0.000	0.005	0.000	0.001
		Variance-Bias Ratio	0.010	0.015	0.609	0.607	1.710	1.814	3.086	3.735	0.809	0.794	0.022	0.028
		Percent Change from Raw	~	~	153.636	153.636	265.145	265.145	440.891	440.891	165.508	165.508	100.728	100.728
	Decision Tree													
		Total Loss	0.061	0.061	0.772	0.772	0.696	0.696	0.684	0.684	0.688	0.688	0.061	0.061
		Bias	0.030	0.036	0.608	0.963	0.491	0.963	0.475	0.963	0.481	0.963	0.030	0.036
		Variance	0.031	0.040	0.164	0.207	0.205	0.289	0.046	0.301	0.207	0.295	0.031	0.040
		Noise	0.000	0.015	0.000	0.398	0.000	0.556	0.162	0.579	0.000	0.569	0.000	0.015
		Variance-Bias Ratio	1.033	1.110	0.269	0.215	0.417	0.300	0.098	0.313	0.430	0.306	1.034	1.111
		Percent Change from Raw	~	~	1259.483	1259.483	1135.652	1135.652	1116.508	1116.508	1123.095	1123.095	100.079	100.079
	Random Forest	**Total Loss**	0.039	0.039	0.822	0.822	0.762	0.762	0.754	0.754	0.749	0.749	0.039	0.039
		Bias	0.031	0.034	0.692	0.963	0.593	0.963	0.581	0.963	0.572	0.963	0.031	0.033
		Variance	0.008	0.011	0.129	0.153	0.169	0.217	0.127	0.225	0.177	0.230	0.008	0.011
		Noise	0.000	0.006	0.000	0.294	0.000	0.417	0.047	0.433	0.000	0.444	0.000	0.005
		Variance-Bias Ratio	0.263	0.336	0.187	0.159	0.285	0.225	0.218	0.234	0.310	0.239	0.263	0.343
		Percent Change from Raw	~	~	2087.923	2087.923	1936.937	1936.937	1916.597	1916.597	1902.651	1902.651	100.093	100.093
	SVM	**Total Loss**	0.038	0.038	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.038	0.038
		Bias	0.035	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.035	0.037
		Variance	0.003	0.004	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.004
		Noise	0.000	0.004	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003
		Variance-Bias Ratio	0.094	0.118	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.086	0.107
		Percent Change from Raw	~	~	97.849	97.849	97.849	97.849	97.849	97.849	97.849	97.849	100.306	100.306
	Gradient Boosting	**Total Loss**	0.046	0.046	0.939	0.939	0.927	0.927	0.934	0.934	0.920	0.920	0.046	0.046
		Bias	0.035	0.042	0.913	0.963	0.891	0.963	0.905	0.963	0.876	0.963	0.035	0.042
		Variance	0.011	0.015	0.026	0.027	0.036	0.038	0.003	0.030	0.044	0.046	0.011	0.015
		Noise	0.000	0.011	0.000	0.051	0.000	0.073	0.027	0.059	0.000	0.089	0.000	0.011
		Variance-Bias Ratio	0.319	0.359	0.028	0.028	0.041	0.039	0.003	0.032	0.050	0.048	0.319	0.360
		Percent Change from Raw	~	~	2032.992	2032.992	2007.584	2007.584	2022.734	2022.734	1991.394	1991.394	100.018	100.018
	Neural Network	**Total Loss**	0.038	0.038	0.406	0.406	0.387	0.387	0.526	0.526	0.178	0.178	0.038	0.038
		Bias	0.037	0.037	0.173	0.137	0.154	0.037	0.279	0.791	0.049	0.037	0.037	0.037
		Variance	0.001	0.001	0.232	0.389	0.234	0.378	0.000	0.460	0.129	0.154	0.001	0.001
		Noise	0.000	0.001	0.000	0.121	0.000	0.028	0.247	0.725	0.000	0.013	0.000	0.001
		Variance-Bias Ratio	0.025	0.031	1.340	2.848	1.521	10.105	0.000	0.582	2.629	4.105	0.017	0.021
		Percent Change from Raw	~	~	1072.621	1072.621	1023.751	1023.751	1391.955	1391.955	471.839	471.839	99.592	99.592

TABLE A.13: Bias-Variance Decomposition Results for Wine Quality Data with Binary Target

Breast Cancer Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile Transform		Quantile Normalize	
Breast cancer data with binary target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.043	0.043	0.626	0.626	0.626	0.626	0.626	0.626	0.626	0.626	0.042	0.042
		Bias	0.029	0.041	0.626	0.626	0.626	0.626	0.626	0.626	0.626	0.626	0.028	0.041
		Variance	0.013	0.019	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.014	0.020
		Noise	0.000	0.017	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.019
		Variance-Bias Ratio	0.451	0.460	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.487	0.485
		Percent Change from Raw	~	~	1465.753	1465.753	1465.753	1465.753	1465.753	1465.753	1465.753	1465.753	98.205	98.205
	Decision Tree													
		Total Loss	0.091	0.090	0.601	0.601	0.514	0.514	0.488	0.488	0.536	0.536	0.090	0.090
		Bias	0.047	0.064	0.566	0.626	0.284	0.620	0.239	0.374	0.322	0.626	0.046	0.064
		Variance	0.043	0.061	0.035	0.037	0.231	0.370	0.370	0.465	0.214	0.313	0.044	0.061
		Noise	0.000	0.036	0.000	0.062	0.000	0.475	0.121	0.352	0.000	0.403	0.000	0.036
		Variance-Bias Ratio	0.911	0.954	0.062	0.059	0.814	0.596	1.544	1.244	0.667	0.500	0.951	0.955
		Percent Change from Raw	~	~	663.908	670.579	568.172	573.881	539.083	544.500	592.106	598.056	99.025	100.020
	Random Forest	**Total Loss**	0.059	0.059	0.626	0.626	0.551	0.551	0.498	0.498	0.590	0.590	0.059	0.059
		Bias	0.048	0.058	0.626	0.626	0.449	0.626	0.289	0.626	0.512	0.626	0.048	0.058
		Variance	0.011	0.015	0.000	0.000	0.102	0.124	0.124	0.313	0.078	0.086	0.011	0.015
		Noise	0.000	0.015	0.000	0.000	0.000	0.199	0.085	0.440	0.000	0.122	0.000	0.015
		Variance-Bias Ratio	0.232	0.259	0.000	0.000	0.226	0.199	0.429	0.500	0.152	0.138	0.232	0.260
		Percent Change from Raw	~	~	1061.508	1061.508	934.306	934.306	845.407	845.407	1000.397	1000.397	100.010	100.010
	SVM	**Total Loss**	0.374	0.374	0.625	0.625	0.422	0.422	0.405	0.405	0.608	0.608	0.374	0.374
		Bias	0.374	0.374	0.622	0.626	0.268	0.374	0.297	0.374	0.543	0.626	0.374	0.374
		Variance	0.000	0.000	0.003	0.003	0.154	0.190	0.190	0.123	0.065	0.070	0.000	0.000
		Noise	0.000	0.000	0.000	0.004	0.000	0.142	0.082	0.092	0.000	0.088	0.000	0.000
		Variance-Bias Ratio	0.000	0.000	0.005	0.005	0.574	0.508	0.639	0.329	0.120	0.112	0.000	0.000
		Percent Change from Raw	~	~	166.986	166.986	112.766	112.766	108.264	108.264	162.484	162.484	100.000	100.000
	Gradient Boosting	**Total Loss**	0.067	0.067	0.624	0.624	0.561	0.561	0.551	0.551	0.596	0.596	0.067	0.067
		Bias	0.042	0.053	0.623	0.626	0.418	0.626	0.343	0.626	0.502	0.626	0.042	0.053
		Variance	0.025	0.034	0.001	0.001	0.143	0.177	0.177	0.295	0.094	0.105	0.025	0.034
		Noise	0.000	0.020	0.000	0.003	0.000	0.241	0.031	0.370	0.000	0.134	0.000	0.020
		Variance-Bias Ratio	0.594	0.653	0.002	0.002	0.341	0.282	0.515	0.471	0.187	0.168	0.594	0.653
		Percent Change from Raw	~	~	938.808	938.808	843.573	843.573	827.809	827.809	896.325	896.325	100.053	100.053
	Neural Network	**Total Loss**	0.069	0.069	0.626	0.626	0.626	0.626	0.626	0.626	0.374	0.374	0.068	0.068
		Bias	0.055	0.058	0.626	0.626	0.626	0.626	0.626	0.626	0.374	0.374	0.055	0.064
		Variance	0.014	0.018	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.016
		Noise	0.000	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012
		Variance-Bias Ratio	0.249	0.305	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.230	0.252
		Percent Change from Raw	~	~	906.933	906.933	906.933	906.933	906.933	906.933	542.761	542.761	98.940	98.940

TABLE A.14: Bias-Variance Decomposition Results for Breast Cancer Data with Binary Target

Voting Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile Transform		Quantile Normalize	
Voting data with binary target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.058	0.058	0.112	0.112	0.058	0.058	0.058	0.058	0.058	0.058	0.059	0.059
		Bias	0.046	0.061	0.078	0.099	0.046	0.061	0.046	0.061	0.046	0.061	0.048	0.061
		Variance	0.011	0.017	0.034	0.045	0.011	0.017	0.017	0.017	0.011	0.017	0.011	0.016
		Noise	0.000	0.020	0.000	0.032	0.000	0.020	0.005	0.020	0.000	0.020	0.000	0.018
		Variance-Bias Ratio	0.247	0.277	0.435	0.449	0.247	0.277	0.365	0.277	0.247	0.277	0.227	0.255
		Percent Change from Raw	~	~	193.123	193.123	100.000	100.000	100.000	100.000	100.000	100.000	101.162	101.162
	Decision Tree													
		Total Loss	0.066	0.066	0.063	0.063	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
		Bias	0.036	0.046	0.040	0.061	0.036	0.046	0.036	0.046	0.036	0.046	0.036	0.046
		Variance	0.030	0.043	0.023	0.035	0.030	0.043	0.043	0.043	0.030	0.043	0.030	0.043
		Noise	0.000	0.023	0.000	0.033	0.000	0.023	0.013	0.023	0.000	0.023	0.000	0.023
		Variance-Bias Ratio	0.853	0.942	0.567	0.577	0.816	0.942	1.182	0.942	0.816	0.942	0.816	0.942
		Percent Change from Raw	~	~	96.100	95.646	100.475	100.000	100.475	100.000	100.475	100.000	100.475	100.000
	Random Forest	**Total Loss**	0.044	0.044	0.053	0.053	0.044	0.044	0.044	0.044	0.044	0.044	0.044	0.044
		Bias	0.031	0.038	0.041	0.046	0.031	0.038	0.031	0.038	0.031	0.038	0.031	0.038
		Variance	0.012	0.018	0.012	0.017	0.012	0.018	0.018	0.018	0.012	0.018	0.012	0.018
		Noise	0.000	0.012	0.000	0.010	0.000	0.012	0.005	0.012	0.000	0.012	0.000	0.012
		Variance-Bias Ratio	0.388	0.462	0.297	0.379	0.388	0.462	0.561	0.462	0.388	0.462	0.388	0.462
		Percent Change from Raw	~	~	121.723	121.723	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	SVM	**Total Loss**	0.052	0.052	0.054	0.054	0.052	0.052	0.052	0.052	0.052	0.052	0.052	0.052
		Bias	0.045	0.053	0.038	0.046	0.045	0.053	0.045	0.053	0.045	0.053	0.045	0.061
		Variance	0.007	0.011	0.016	0.023	0.007	0.011	0.011	0.011	0.007	0.011	0.007	0.011
		Noise	0.000	0.012	0.000	0.015	0.000	0.012	0.004	0.012	0.000	0.012	0.000	0.020
		Variance-Bias Ratio	0.162	0.208	0.427	0.510	0.162	0.208	0.247	0.208	0.162	0.208	0.163	0.184
		Percent Change from Raw	~	~	102.883	102.883	100.000	100.000	100.000	100.000	100.000	100.000	99.649	99.649
	Gradient Boosting	**Total Loss**	0.056	0.056	0.058	0.058	0.056	0.056	0.056	0.056	0.056	0.056	0.056	0.056
		Bias	0.033	0.046	0.039	0.046	0.033	0.046	0.033	0.046	0.033	0.046	0.033	0.046
		Variance	0.022	0.032	0.019	0.028	0.022	0.032	0.032	0.032	0.022	0.032	0.022	0.032
		Noise	0.000	0.022	0.000	0.016	0.000	0.022	0.009	0.022	0.000	0.022	0.000	0.022
		Variance-Bias Ratio	0.668	0.693	0.472	0.617	0.668	0.693	0.949	0.693	0.668	0.693	0.668	0.693
		Percent Change from Raw	~	~	103.490	103.490	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Neural Network	**Total Loss**	0.059	0.059	0.079	0.079	0.059	0.059	0.059	0.059	0.059	0.059	0.058	0.058
		Bias	0.045	0.053	0.059	0.076	0.045	0.053	0.045	0.053	0.045	0.053	0.044	0.053
		Variance	0.014	0.020	0.021	0.028	0.014	0.020	0.020	0.020	0.014	0.020	0.014	0.019
		Noise	0.000	0.014	0.000	0.025	0.000	0.014	0.006	0.014	0.000	0.014	0.000	0.014
		Variance-Bias Ratio	0.312	0.372	0.353	0.362	0.312	0.372	0.442	0.372	0.312	0.372	0.307	0.348
		Percent Change from Raw	~	~	134.324	134.324	100.000	100.000	100.000	100.000	100.000	100.000	98.049	98.049

TABLE A.15: Bias-Variance Decomposition Results for Congressional Voting Data with Binary Target

Abalone Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile Transform		Quantile Normalize	
Abalone data with binary target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093
		Bias	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093
		Variance	0.000	0.000	0.000	0.000	0.000	0.000	0.144	0.000	0.000	0.000	0.000	0.000
		Noise	0.000	0.000	0.000	0.000	0.000	0.000	0.144	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.000	0.000	0.000	0.000	0.000	0.000	1.542	0.000	0.000	0.000	0.000	0.000
		Percent Change from Raw	~	~	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Decision Tree													
		Total Loss	0.145	0.145	0.102	0.102	0.230	0.230	0.252	0.252	0.391	0.391	0.145	0.145
		Bias	0.079	0.108	0.091	0.093	0.098	0.133	0.111	0.156	0.174	0.160	0.079	0.108
		Variance	0.066	0.097	0.011	0.012	0.131	0.197	0.046	0.206	0.217	0.374	0.066	0.097
		Noise	0.000	0.060	0.000	0.004	0.000	0.101	0.095	0.110	0.000	0.143	0.000	0.060
		Variance-Bias Ratio	0.847	0.892	0.124	0.131	1.340	1.482	0.419	1.318	1.248	2.331	0.847	0.892
		Percent Change from Raw	~	~	70.193	70.193	158.349	158.349	173.704	173.704	269.903	269.903	100.000	100.000
	Random Forest	**Total Loss**	0.108	0.108	0.093	0.093	0.122	0.122	0.127	0.127	0.149	0.149	0.108	0.108
		Bias	0.081	0.100	0.093	0.093	0.077	0.093	0.082	0.093	0.079	0.093	0.081	0.100
		Variance	0.027	0.037	0.000	0.000	0.045	0.056	0.127	0.055	0.070	0.079	0.027	0.037
		Noise	0.000	0.030	0.000	0.000	0.000	0.027	0.081	0.021	0.000	0.023	0.000	0.030
		Variance-Bias Ratio	0.337	0.371	0.000	0.000	0.587	0.597	1.554	0.586	0.881	0.851	0.337	0.371
		Percent Change from Raw	~	~	86.254	86.254	112.596	112.596	117.182	117.182	138.182	138.182	100.000	100.000
	SVM	**Total Loss**	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093
		Bias	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093
		Variance	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Noise	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Percent Change from Raw	~	~	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Gradient Boosting	**Total Loss**	0.103	0.103	0.094	0.094	0.171	0.171	0.213	0.213	0.237	0.237	0.103	0.103
		Bias	0.081	0.093	0.093	0.093	0.086	0.117	0.109	0.166	0.105	0.114	0.081	0.093
		Variance	0.021	0.028	0.001	0.001	0.085	0.120	0.003	0.150	0.132	0.184	0.021	0.028
		Noise	0.000	0.018	0.000	0.000	0.000	0.066	0.102	0.102	0.000	0.061	0.000	0.018
		Variance-Bias Ratio	0.261	0.296	0.015	0.015	0.985	1.023	0.025	0.902	1.263	1.616	0.261	0.296
		Percent Change from Raw	~	~	92.110	92.110	166.500	166.500	207.969	207.969	230.980	230.980	100.000	100.000
	Neural Network	**Total Loss**	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093
		Bias	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093	0.093
		Variance	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Noise	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.004	0.001	0.001
		Percent Change from Raw	~	~	99.989	99.989	99.985	99.985	99.987	99.987	100.166	100.166	100.055	100.055

TABLE A.16: Bias-Variance Decomposition Results for Abalone Data with Binary Target

Arrhythmia Data with Binary Target

Data	Model	Normalization	None		Z-standard		Min-Max		MaxAbs (-1,1)		Quantile Transform		Quantile Normalize	
Arrhythmia data with binary target	Logistic	Type of Loss	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1	MSE	0-1
		Total Loss	0.065	0.065	0.875	0.875	0.755	0.755	0.636	0.636	0.905	0.905	0.064	0.064
		Bias	0.046	0.051	0.828	0.904	0.650	0.838	0.476	0.713	0.877	0.934	0.046	0.051
		Variance	0.018	0.024	0.047	0.059	0.105	0.144	0.144	0.236	0.027	0.034	0.018	0.024
		Noise	0.000	0.010	0.000	0.089	0.000	0.227	0.016	0.313	0.000	0.063	0.000	0.011
		Variance-Bias Ratio	0.398	0.459	0.057	0.065	0.161	0.172	0.302	0.331	0.031	0.036	0.395	0.457
		Percent Change from Raw	~	~	1350.982	1350.982	1165.815	1165.815	983.032	983.032	1397.399	1397.399	98.683	98.683
	Decision Tree													
		Total Loss	0.034	0.034	0.094	0.094	0.099	0.099	0.098	0.098	0.100	0.100	0.034	0.034
		Bias	0.018	0.022	0.056	0.059	0.055	0.059	0.056	0.059	0.055	0.059	0.018	0.022
		Variance	0.016	0.024	0.038	0.040	0.044	0.046	0.046	0.044	0.044	0.047	0.016	0.024
		Noise	0.000	0.011	0.000	0.005	0.000	0.006	0.004	0.005	0.000	0.006	0.000	0.011
		Variance-Bias Ratio	0.907	1.071	0.691	0.682	0.796	0.790	0.834	0.754	0.798	0.791	0.907	1.071
		Percent Change from Raw	~	~	274.608	274.608	290.841	290.841	286.111	286.111	291.056	291.056	100.000	100.000
	Random Forest	**Total Loss**	0.059	0.059	0.111	0.111	0.172	0.172	0.153	0.153	0.160	0.160	0.059	0.059
		Bias	0.059	0.059	0.057	0.059	0.063	0.059	0.062	0.059	0.062	0.059	0.059	0.059
		Variance	0.001	0.001	0.054	0.058	0.108	0.127	0.127	0.105	0.098	0.112	0.001	0.001
		Noise	0.000	0.000	0.000	0.006	0.000	0.014	0.035	0.010	0.000	0.012	0.000	0.000
		Variance-Bias Ratio	0.013	0.014	0.937	0.980	1.708	2.153	2.057	1.782	1.594	1.911	0.013	0.014
		Percent Change from Raw	~	~	186.609	186.609	289.307	289.307	258.218	258.218	268.824	268.824	100.000	100.000
	SVM	**Total Loss**	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059
		Bias	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059
		Variance	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Noise	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Variance-Bias Ratio	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		Percent Change from Raw	~	~	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
	Gradient Boosting	**Total Loss**	0.033	0.033	0.061	0.061	0.061	0.061	0.061	0.061	0.061	0.061	0.033	0.033
		Bias	0.019	0.022	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.019	0.022
		Variance	0.014	0.022	0.002	0.002	0.003	0.003	0.003	0.003	0.003	0.003	0.014	0.022
		Noise	0.000	0.012	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012
		Variance-Bias Ratio	0.768	1.018	0.036	0.036	0.046	0.046	0.046	0.048	0.047	0.047	0.768	1.018
		Percent Change from Raw	~	~	184.039	184.039	185.689	185.689	185.823	185.823	185.800	185.800	100.000	100.000
	Neural Network	**Total Loss**	0.071	0.071	0.387	0.387	0.059	0.059	0.061	0.061	0.074	0.074	0.073	0.073
		Bias	0.057	0.051	0.206	0.331	0.059	0.059	0.058	0.059	0.057	0.059	0.057	0.059
		Variance	0.014	0.020	0.181	0.279	0.000	0.000	0.000	0.003	0.018	0.018	0.015	0.022
		Noise	0.000	0.001	0.000	0.223	0.000	0.000	0.003	0.000	0.000	0.002	0.000	0.008
		Variance-Bias Ratio	0.251	0.387	0.880	0.843	0.000	0.000	0.000	0.044	0.309	0.303	0.268	0.370
		Percent Change from Raw	~	~	547.100	547.100	83.160	83.160	86.268	86.268	104.927	104.927	102.682	102.682

TABLE A.17: Bias-Variance Decomposition Results for Arrhythmia Data with Binary Target

Forest Fires Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile Transform	Quantile Normalize
Forest Fires Data - Continuous Target	Linear	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	8254.563	4636426.991	54418873.320	61656927.105	77782033.117	8254.563
		Bias	8188.901	3419009.689	38494821.256	43230677.674	12450153.080	8188.901
		Variance	65.662	1217417.301	15924052.064	15924052.064	65331880.037	65.662
		Noise	0.000	0.000	0.000	2502197.367	0.000	0.000
		Variance-Bias Ratio	0.008	0.356	0.414	0.368	5.247	0.008
		Percent Change from Raw	~	56168.051	659258.104	746943.595	942291.388	100.000
	Decision Tree							
		Total Loss	14564.197	174310.342	191218.064	191189.902	191069.504	14564.197
		Bias	9705.018	67683.825	82269.473	82240.074	82166.683	9705.018
		Variance	4859.179	106626.517	108948.592	108948.592	108902.820	4859.179
		Noise	0.000	0.000	0.000	1.237	0.000	0.000
		Bias-Variance Ratio	0.501	1.575	1.324	1.325	1.325	0.501
		Percent Change from Raw	~	1196.841	1312.932	1312.739	1311.912	100.000
	Random Forest	**Total Loss**	9973.155	70512.566	83921.663	83911.156	83802.232	9973.155
		Bias	9118.659	48292.165	60221.253	60210.717	60139.045	9118.659
		Variance	854.497	22220.400	23700.410	23700.410	23663.187	854.497
		Noise	0.000	0.000	0.000	0.029	0.000	0.000
		Bias-Variance Ratio	0.094	0.460	0.394	0.394	0.393	0.094
		Percent Change from Raw	~	707.024	841.476	841.370	840.278	100.000
	SVM	**Total Loss**	8527.384	8520.206	8540.229	8542.123	8535.867	8527.384
		Bias	8527.287	8520.015	8539.941	8541.850	8535.546	8527.287
		Variance	0.096	0.191	0.288	0.288	0.321	0.096
		Noise	0.000	0.000	0.000	0.014	0.000	0.000
		Bias-Variance Ratio	0.000	0.000	0.000	0.000	0.000	0.000
		Percent Change from Raw	~	99.916	100.151	100.173	100.099	100.000
	Gradient Boosting	**Total Loss**	12412.847	152934.223	173769.259	173744.792	173578.379	12412.847
		Bias	9842.247	87776.569	108617.796	108588.723	108475.548	9842.247
		Variance	2570.601	65157.654	65151.463	65151.463	65102.831	2570.601
		Noise	0.000	0.000	0.000	4.608	0.000	0.000
		Bias-Variance Ratio	0.261	0.742	0.600	0.600	0.600	0.261
		Percent Change from Raw	~	1232.064	1399.915	1399.717	1398.377	100.000
	Neural Network	**Total Loss**	8253.932	4636990.709	54421909.843	61660412.813	77784164.639	8253.828
		Bias	8190.282	3419289.295	38495662.256	43234843.780	12450637.226	8190.201
		Variance	63.650	1217701.414	15926247.585	15926247.585	65333527.413	63.621
		Noise	0.000	0.000	0.000	2499321.449	0.000	0.000
		Bias-Variance Ratio	0.008	0.356	0.414	0.368	5.247	0.008
		Percent Change from Raw	~	56179.172	659345.259	747042.891	942389.202	99.999

TABLE A.18: Bias-Variance Decomposition Results for Forest Fires Data with Continuous Target

Solar Flares Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile Transform	Quantile Normalize
Solar Flare Data - Continuous Target	Linear	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	0.376	0.457	0.376	0.376	0.376	0.376
		Bias	0.346	0.454	0.345	0.346	0.346	0.345
		Variance	0.030	0.002	0.031	0.031	0.030	0.031
		Noise	0.000	0.000	0.000	0.001	0.000	0.000
		Variance-Bias Ratio	0.087	0.005	0.089	0.089	0.087	0.089
		Percent Change from Raw	~	121.569	100.043	100.000	100.000	100.127
	Decision Tree							
		Total Loss	0.769	0.734	0.769	0.769	0.769	0.772
		Bias	0.534	0.408	0.534	0.534	0.534	0.534
		Variance	0.235	0.326	0.235	0.235	0.235	0.237
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Bias-Variance Ratio	0.440	0.799	0.440	0.440	0.440	0.444
		Percent Change from Raw	~	95.427	100.000	100.000	100.000	100.413
	Random Forest	**Total Loss**	0.408	0.459	0.408	0.408	0.408	0.408
		Bias	0.380	0.459	0.380	0.380	0.380	0.380
		Variance	0.028	0.000	0.028	0.028	0.028	0.028
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Bias-Variance Ratio	0.074	0.000	0.074	0.074	0.074	0.074
		Percent Change from Raw	~	112.494	100.000	100.000	100.000	99.989
	SVM	**Total Loss**	0.457	0.459	0.457	0.457	0.457	0.457
		Bias	0.454	0.459	0.455	0.454	0.455	0.455
		Variance	0.003	0.000	0.002	0.002	0.002	0.003
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Bias-Variance Ratio	0.006	0.000	0.005	0.005	0.005	0.006
		Percent Change from Raw	~	100.520	100.094	100.000	100.094	100.018
	Gradient Boosting	**Total Loss**	0.475	0.495	0.474	0.475	0.475	0.474
		Bias	0.386	0.452	0.386	0.386	0.386	0.386
		Variance	0.089	0.043	0.089	0.089	0.089	0.089
		Noise	0.000	0.000	0.000	0.000	0.000	0.000
		Bias-Variance Ratio	0.230	0.095	0.230	0.230	0.230	0.230
		Percent Change from Raw	~	104.234	99.950	100.000	100.019	99.971
	Neural Network	**Total Loss**	0.376	0.459	0.382	0.376	0.380	0.376
		Bias	0.346	0.454	0.343	0.346	0.343	0.346
		Variance	0.030	0.006	0.039	0.039	0.037	0.030
		Noise	0.000	0.000	0.000	0.009	0.000	0.000
		Bias-Variance Ratio	0.087	0.012	0.114	0.114	0.107	0.087
		Percent Change from Raw	~	122.285	101.825	100.017	101.063	100.011

TABLE A.19: Bias-Variance Decomposition Results for Solar Flares Data with Continuous Target

Auto MPG Data with Continuous Target

Data	Model	Normalization	None	Z-standard	Min-Max	MaxAbs (-1,1)	Quantile Transform	Quantile Normalize
Auto mpg Data - Continuous Target	Linear	Type of Loss	MSE	MSE	MSE	MSE	MSE	MSE
		Total Loss	12.98046102	29454928	21.07621525	10679343815	133029193	12.98015591
		Bias	12.26544827	289704291.2	15.90522095	10498618797	1271499879	12.26513305
		Variance	0.715012746	4844995.802	5.170994305	5.170994305	58792054.59	0.715022883
		Noise	0.000	0.000	0.000	180725013.172	0.000	0.000
		Variance-Bias Ratio	0.058	0.017	0.325	0.000	0.046	0.058
		Percent Change from Raw	~	2269174312.476	162.369	82272453970.192	10248418231.961	99.998
	Decision Tree							
		Total Loss	20.35952034	192.7351085	192.3863593	192.6548085	192.6548085	20.34683051
		Bias	11.75359202	190.2749575	189.3929514	189.6552294	189.6552294	11.73801808
		Variance	8.605928322	2.460151	2.993407966	2.993407966	2.999579076	8.608812431
		Noise	0.000	0.000	0.000	0.006	0.000	0.000
		Bias-Variance Ratio	0.732	0.013	0.016	0.016	0.016	0.733
		Percent Change from Raw	~	946.658	944.945	946.264	946.264	99.938
	Random Forest	**Total Loss**	13.88021356	198.0505356	196.9833525	196.9741881	196.9799559	13.87697119
		Bias	12.26761708	197.5658196	196.3927268	196.3792721	196.3848908	12.26589744
		Variance	1.612596483	0.484716	0.59062578	0.59062578	0.595065085	1.611073784
		Noise	0.000	0.000	0.000	0.004	0.000	0.000
		Bias-Variance Ratio	0.131	0.002	0.003	0.003	0.003	0.131
		Percent Change from Raw	~	1426.855	1419.167	1419.101	1419.142	99.977
	SVM	**Total Loss**	64.88569492	65.64744407	66.1892	65.76787458	65.50859322	64.88569492
		Bias	64.42644	65.51012007	65.930898	65.32432358	65.23181822	64.42644
		Variance	0.459254915	0.137324	0.258304	0.258304	0.276775	0.459254915
		Noise	0.000	0.000	0.000	0.185	0.000	0.000
		Bias-Variance Ratio	0.007	0.002	0.004	0.004	0.004	0.007
		Percent Change from Raw	~	101.174	102.009	101.360	100.960	100.000
	Gradient Boosting	**Total Loss**	12.87477458	165.7410966	147.1753051	147.0851763	147.0084356	12.87114746
		Bias	9.470313237	158.7233288	138.9423038	138.8274936	138.7787025	9.467715856
		Variance	3.404461339	7.017767822	8.233001305	8.233001305	8.22973311	3.403431607
		Noise	0.000	0.000	0.000	0.029	0.000	0.000
		Bias-Variance Ratio	0.359	0.044	0.059	0.059	0.059	0.359
		Percent Change from Raw	~	1287.332	1143.129	1142.425	1141.833	99.972
	Neural Network	**Total Loss**	19.39505593	295027446.4	5028226627	10679463722	1330243795	19.17344576
		Bias	16.20878464	290183140.4	4945080967	10498761102	1271459586	16.07586741
		Variance	3.186271297	4844306.03	83145660.64	83145660.64	58784209.06	3.097578291
		Noise	0.000	0.000	0.000	97556959.755	0.000	0.000
		Bias-Variance Ratio	0.197	0.017	0.017	0.008	0.046	0.193
		Percent Change from Raw	~	1521147695.846	25925300988.752	55062814767.424	6858674704.191	98.857

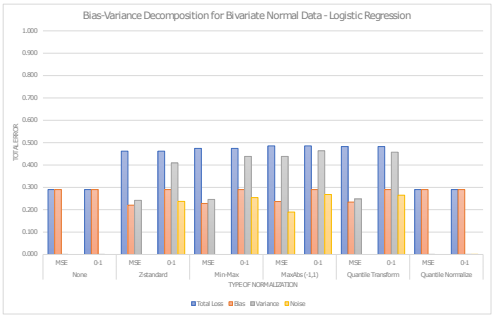
TABLE A.20: Bias-Variance Decomposition Results for Auto MPG Data with Continuous Target

Appendix B

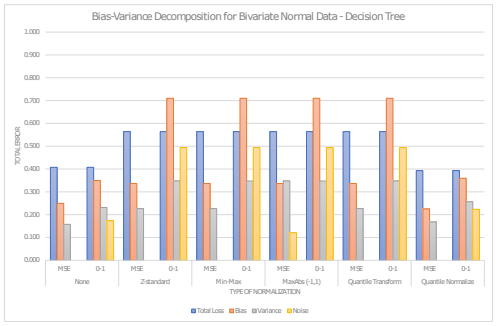
Figures

B.0.1 Simulations

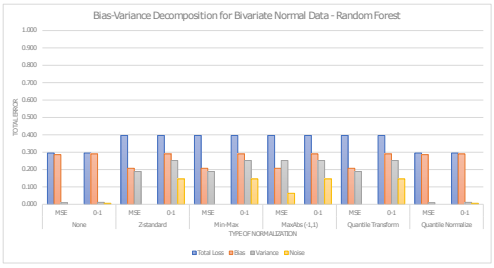
Bivariate Normal Data with Binary Target



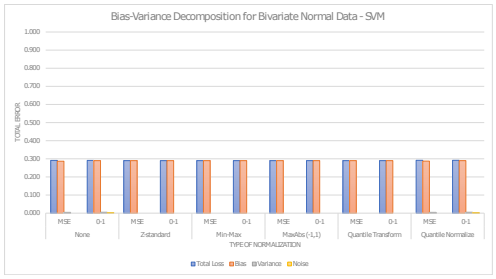
(A) Logistic Regression



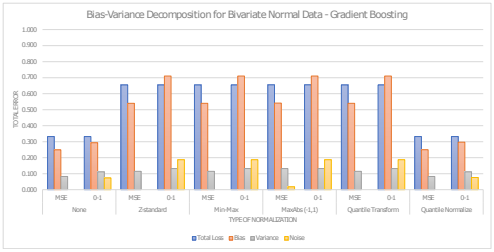
(B) Decision Tree



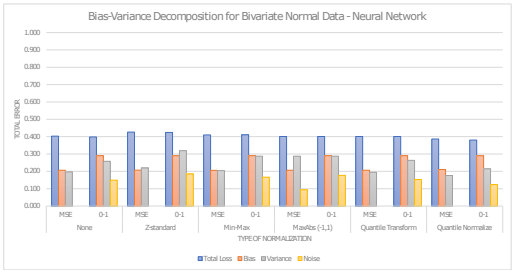
(C) Random Forest



(D) Support Vector Machine (SVM)



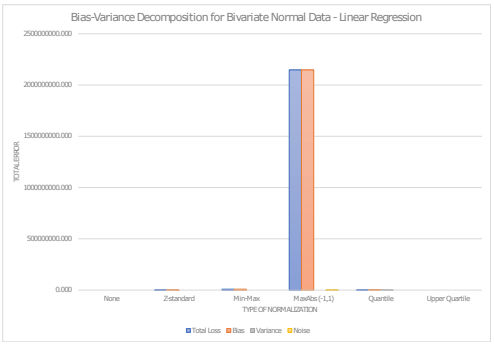
(E) Gradient Boosting Regression (GBR)



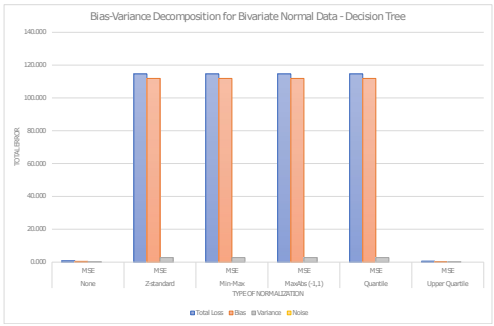
(F) Neural Network

FIGURE B.1: Bias-Variance Decomposition for Bivariate Normal Data with Binary Target

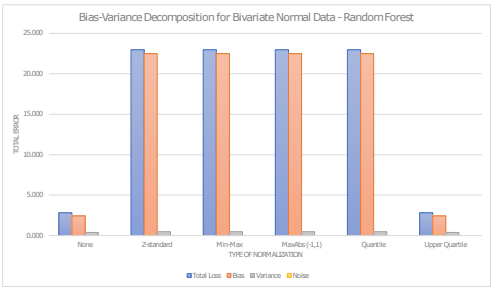
Bivariate Normal Data with Continuous Target



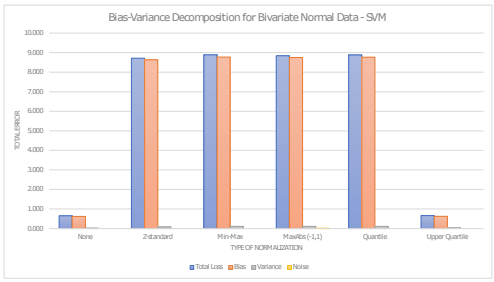
(A) Linear Regression



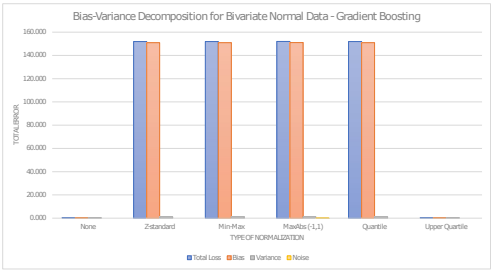
(B) Decision Tree



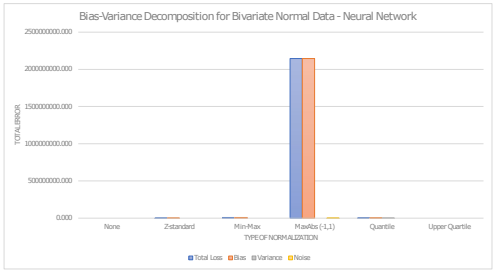
(C) Random Forest



(D) Support Vector Machine (SVM)



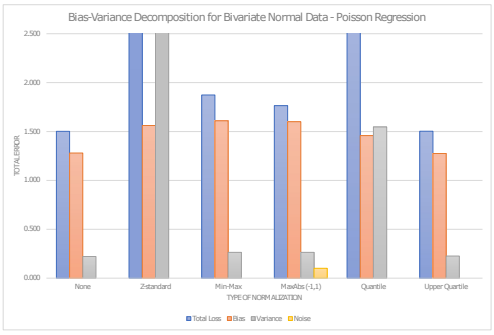
(E) Gradient Boosting Regression (GBR)



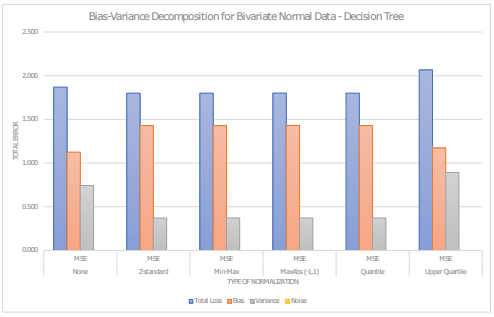
(F) Neural Network

FIGURE B.2: Bias-Variance Decomposition for Bivariate Normal Data with Continuous Target

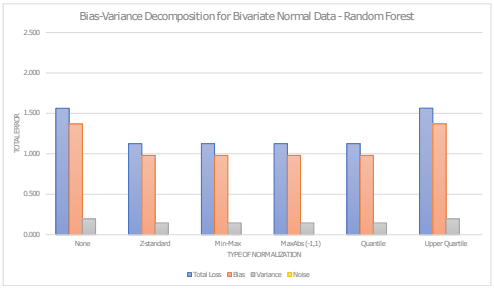
Bivariate Normal Data with Poisson Target



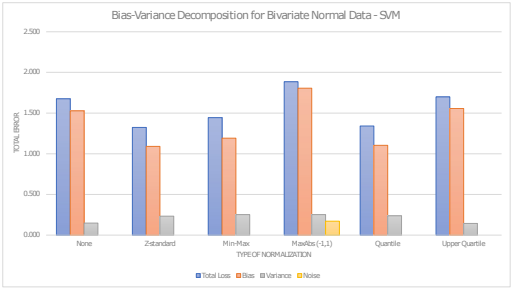
(A) Poisson Regression



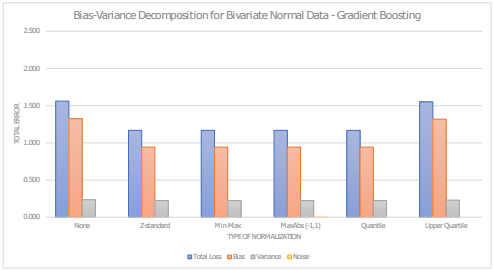
(B) Decision Tree



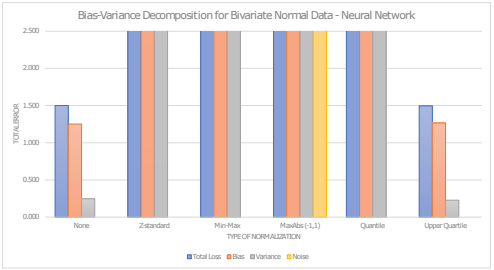
(C) Random Forest



(D) Support Vector Machine (SVM)



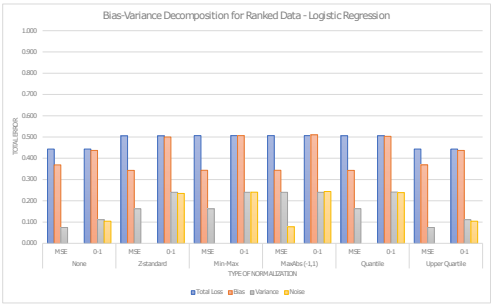
(E) Gradient Boosting Regression (GBR)



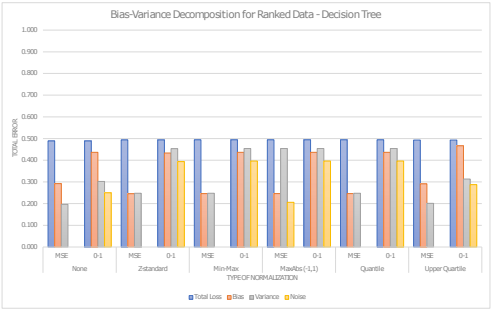
(F) Neural Network

FIGURE B.3: Bias-Variance Decomposition for Bivariate Normal Data with Poisson Target

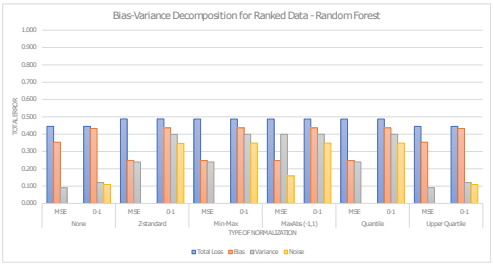
Ranked Data with Binary Target



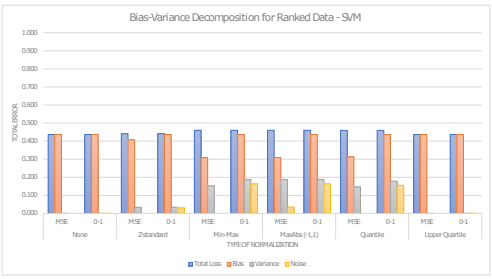
(A) Logistic Regression



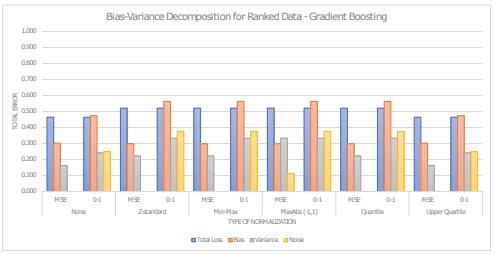
(B) Decision Tree



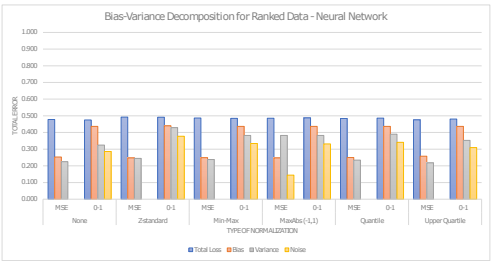
(C) Random Forest



(D) Support Vector Machine (SVM)



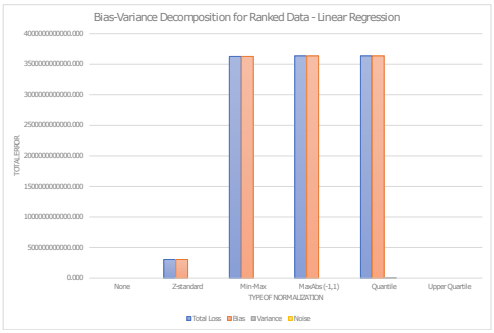
(E) Gradient Boosting Regression (GBR)



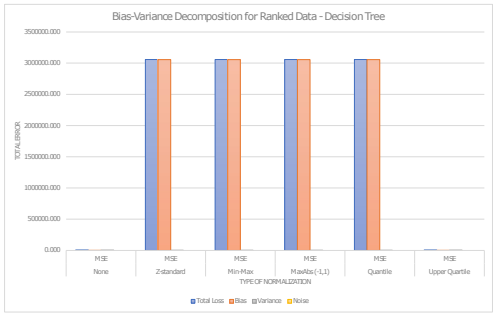
(F) Neural Network

FIGURE B.4: Bias-Variance Decomposition for Ranked Data with Binary Target

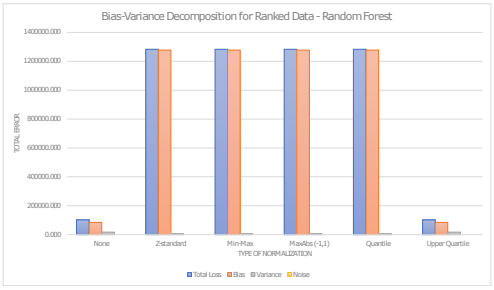
Ranked Data with Continuous Target



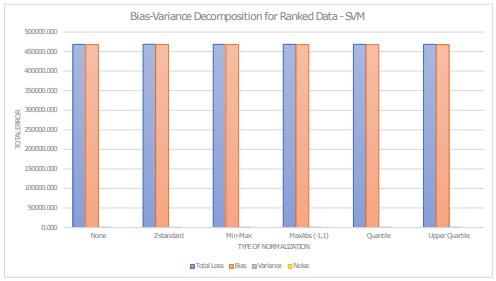
(A) Linear Regression



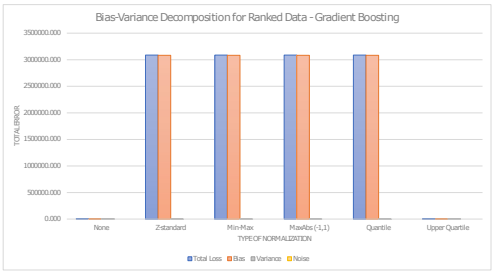
(B) Decision Tree



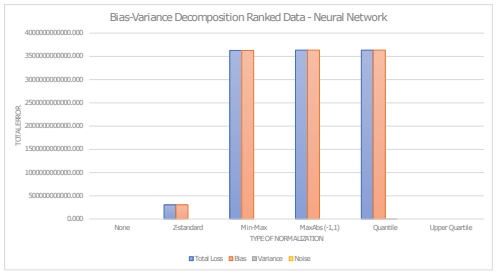
(C) Random Forest



(D) Support Vector Machine (SVM)



(E) Gradient Boosting Regression (GBR)

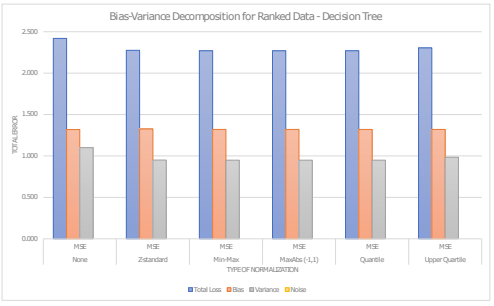


(F) Neural Network

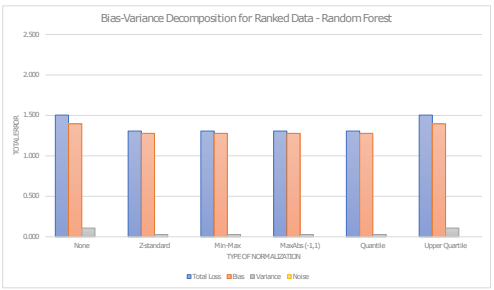
FIGURE B.5: Bias-Variance Decomposition for Ranked Data with Continuous Target

Ranked Data with Poisson Target

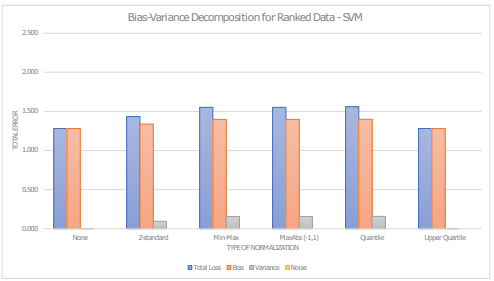
(A) Poisson Regression



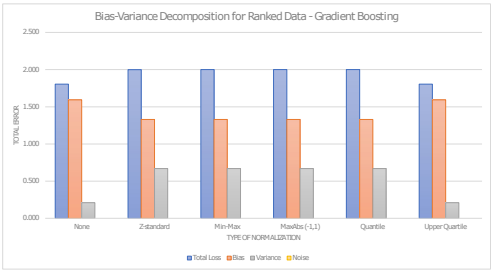
(B) Decision Tree



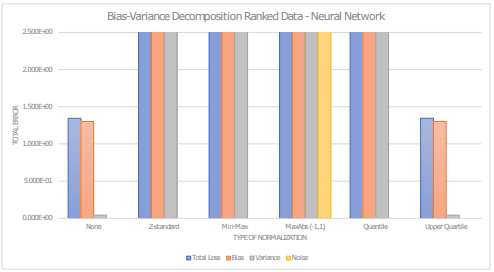
(C) Random Forest



(D) Support Vector Machine (SVM)



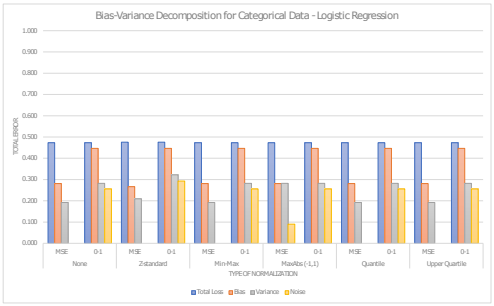
(E) Gradient Boosting Regression (GBR)



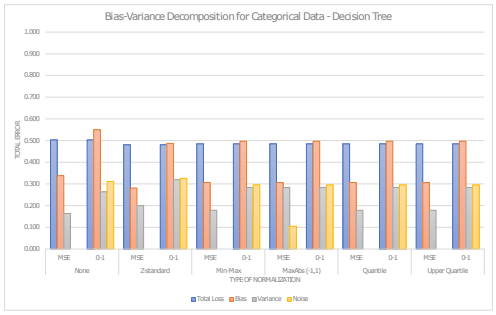
(F) Neural Network

FIGURE B.6: Bias-Variance Decomposition for Ranked Data with Poisson Target

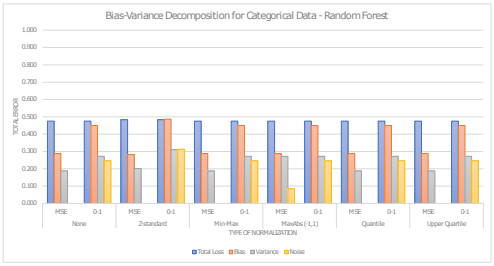
Categorical Data with Binary Target



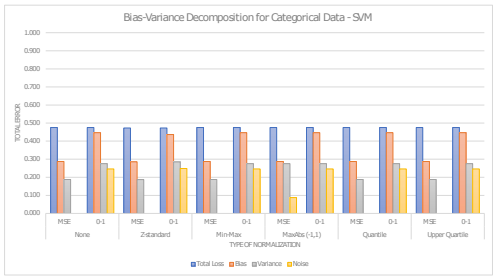
(A) Logistic Regression



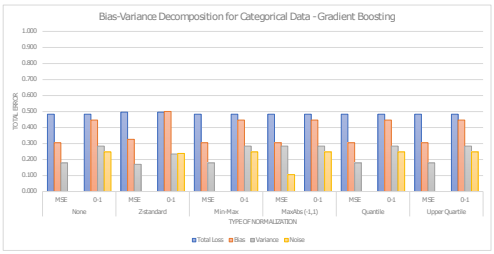
(B) Decision Tree



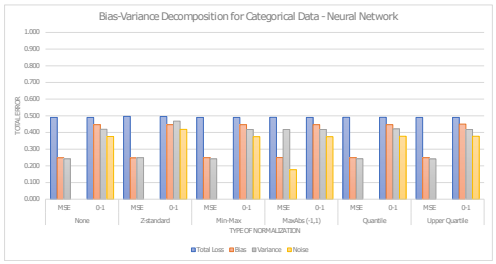
(C) Random Forest



(D) Support Vector Machine (SVM)



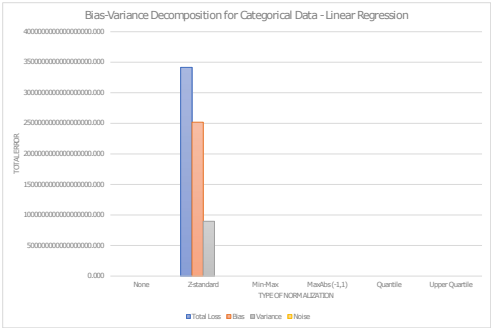
(E) Gradient Boosting Regression (GBR)



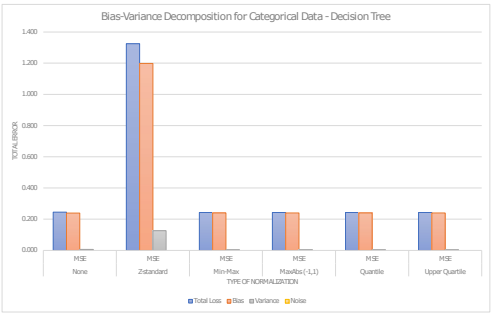
(F) Neural Network

FIGURE B.7: Bias-Variance Decomposition for Categorical Data with Binary Target

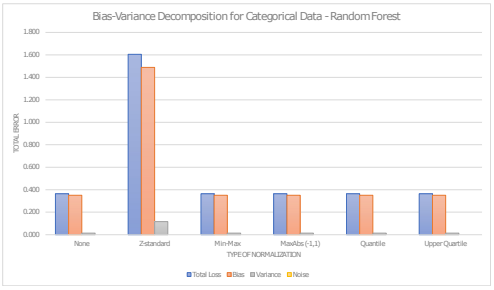
Categorical Data with Continuous Target



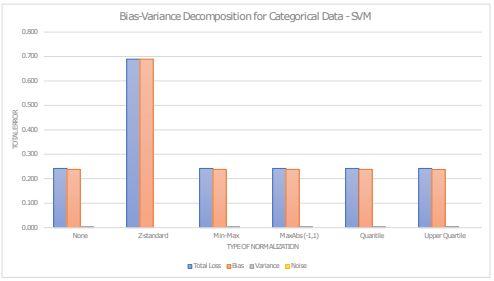
(A) Linear Regression



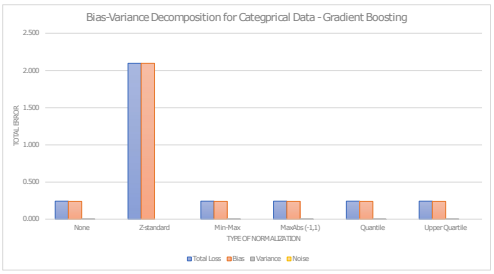
(B) Decision Tree



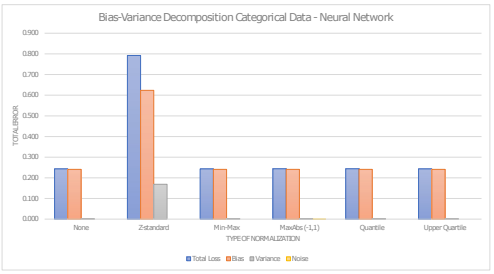
(C) Random Forest



(D) Support Vector Machine (SVM)



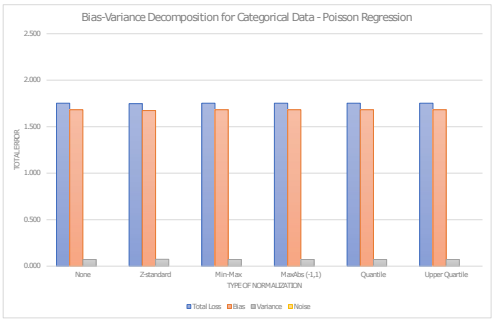
(E) Gradient Boosting Regression (GBR)



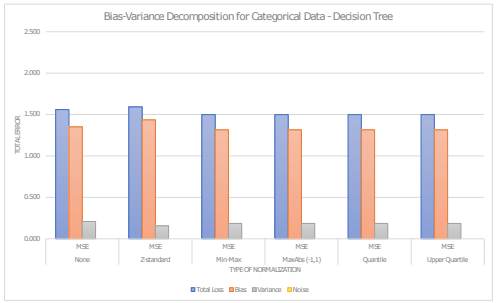
(F) Neural Network

FIGURE B.8: Bias-Variance Decomposition for Categorical Data with Continuous Target

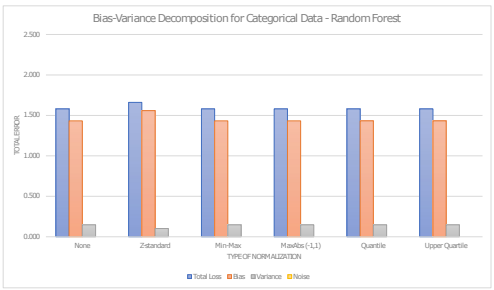
Categorical Data with Poisson Target



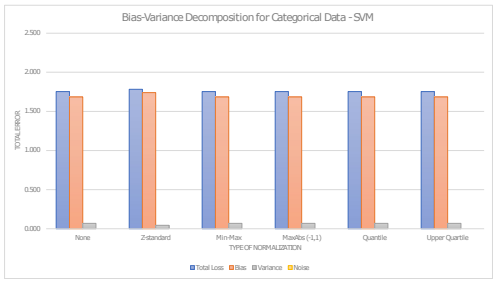
(A) Poisson Regression



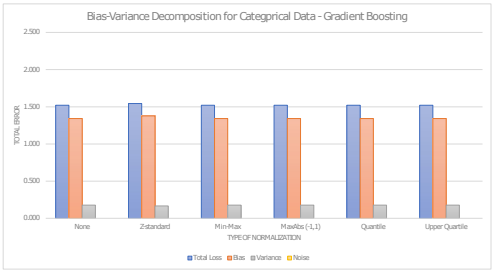
(B) Decision Tree



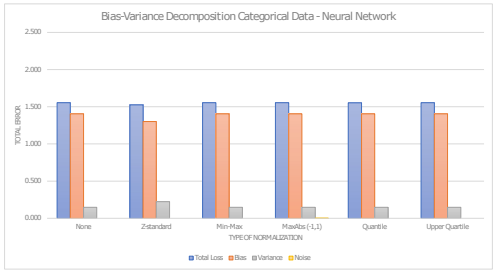
(C) Random Forest



(D) Support Vector Machine (SVM)



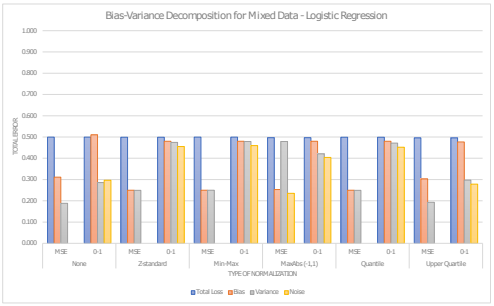
(E) Gradient Boosting Regression (GBR)



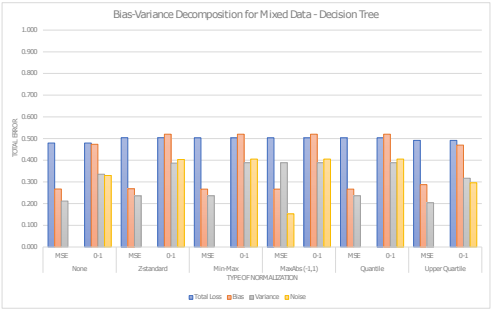
(F) Neural Network

FIGURE B.9: Bias-Variance Decomposition for Categorical Data with Poisson Target

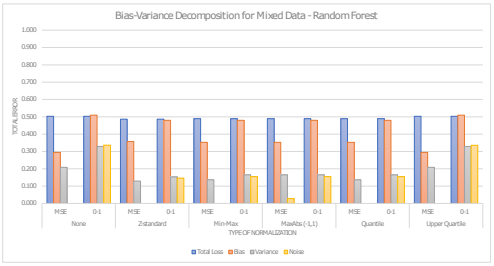
Mixed Data with Binary Target



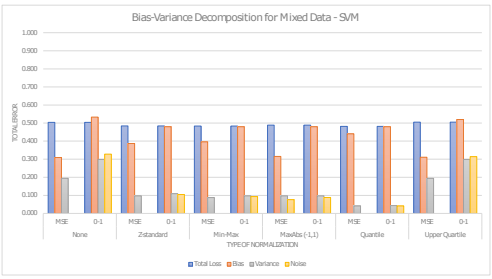
(A) Logistic Regression



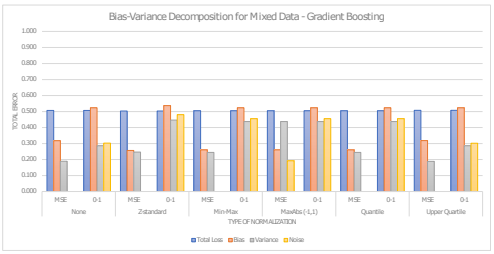
(B) Decision Tree



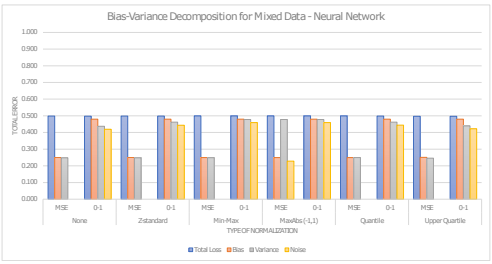
(C) Random Forest



(D) Support Vector Machine (SVM)



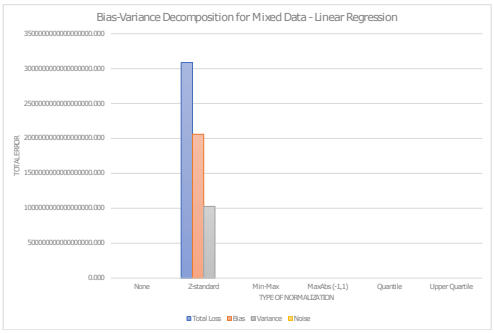
(E) Gradient Boosting Regression (GBR)



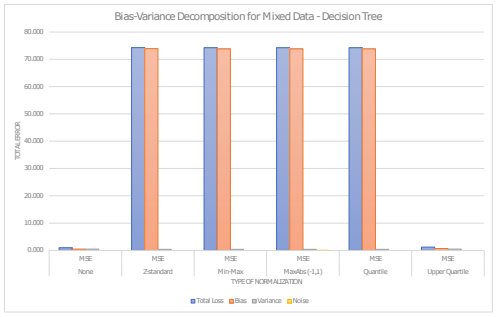
(F) Neural Network

FIGURE B.10: Bias-Variance Decomposition for Mixed Data with Binary Target

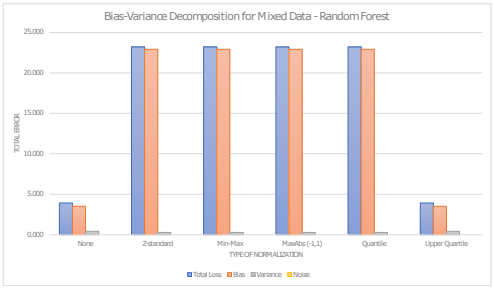
Mixed Data with Continuous Target



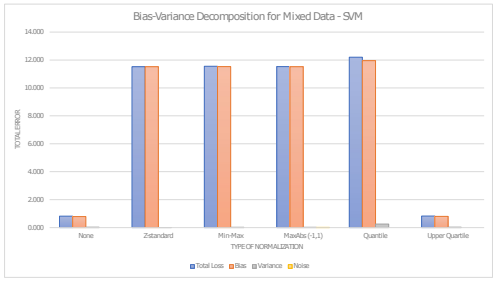
(A) Linear Regression



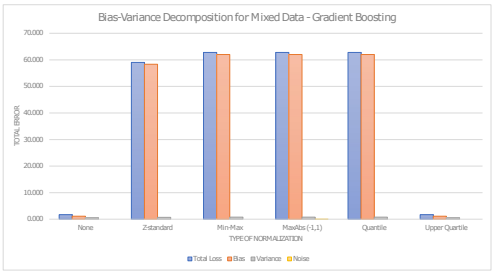
(B) Decision Tree



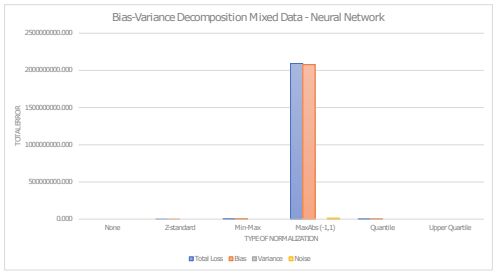
(C) Random Forest



(D) Support Vector Machine (SVM)



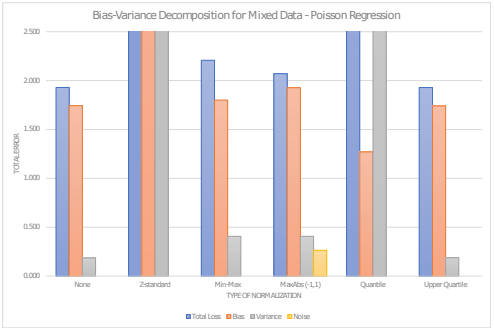
(E) Gradient Boosting Regression (GBR)



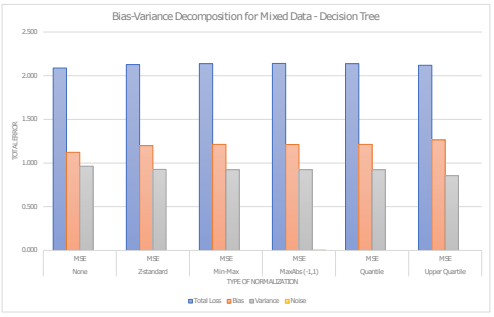
(F) Neural Network

FIGURE B.11: Bias-Variance Decomposition for Mixed Data with Continuous Target

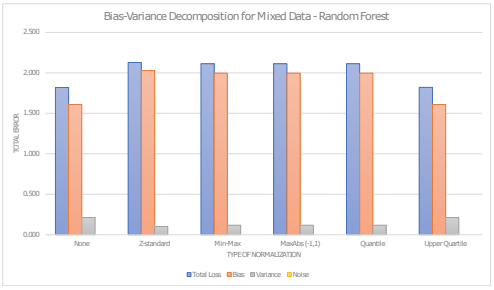
Mixed Data with Poisson Target



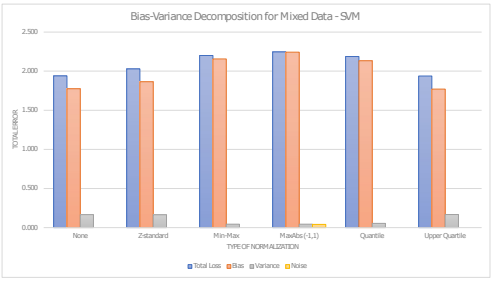
(A) Poisson Regression



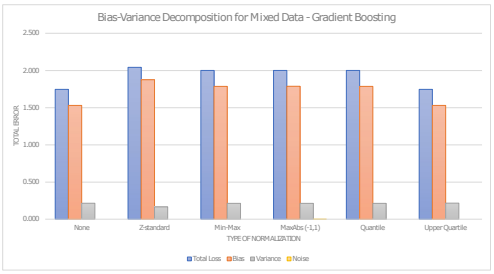
(B) Decision Tree



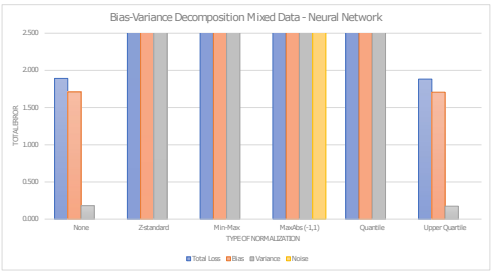
(C) Random Forest



(D) Support Vector Machine (SVM)



(E) Gradient Boosting Regression (GBR)

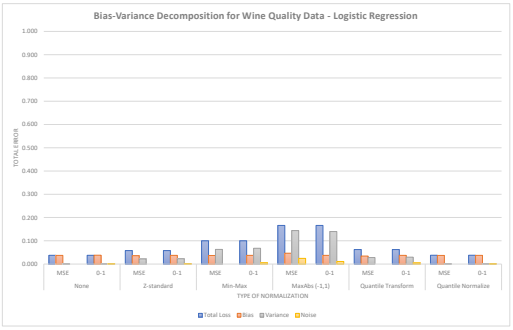


(F) Neural Network

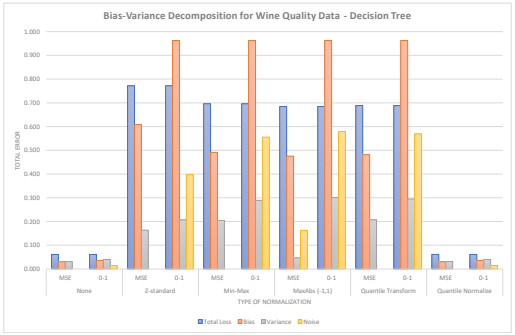
FIGURE B.12: Bias-Variance Decomposition for Mixed Data with Poisson Target

B.0.2 Benchmark Data Results

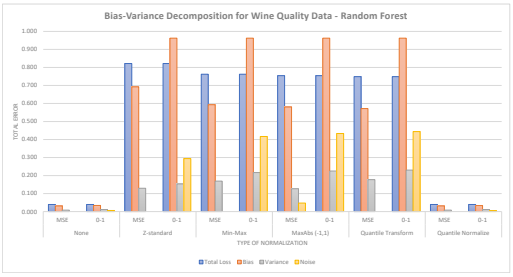
Wine Quality Data



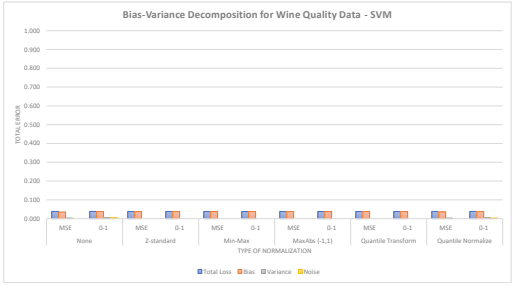
(A) Logistic Regression



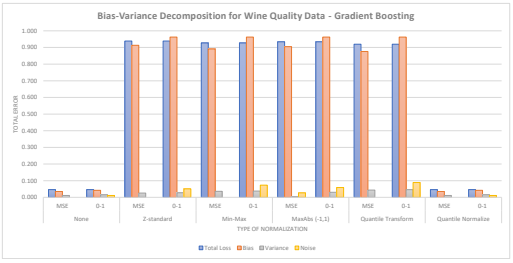
(B) Decision Tree



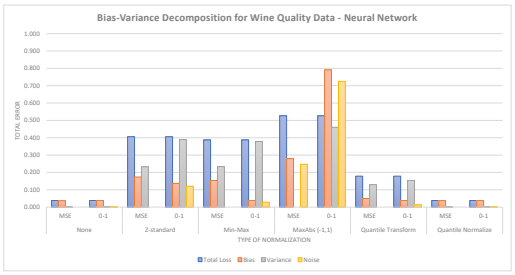
(C) Random Forest



(D) Support Vector Machine (SVM)



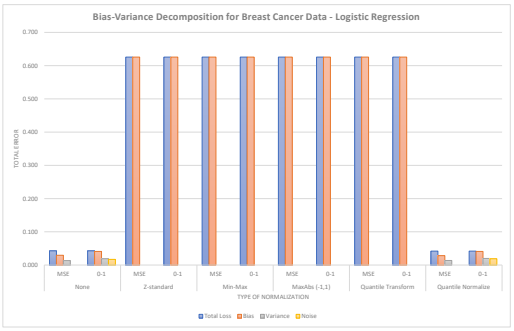
(E) Gradient Boosting Regression (GBR)



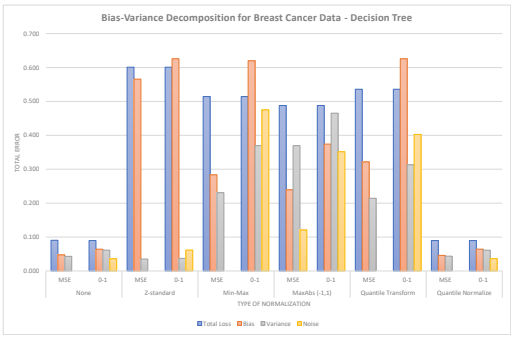
(F) Neural Network

FIGURE B.13: Bias-Variance Decomposition for Wine Quality Data with Binary Target

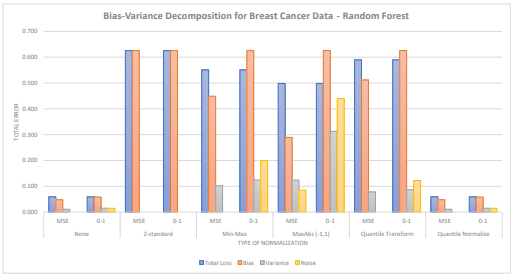
Breast Cancer Data



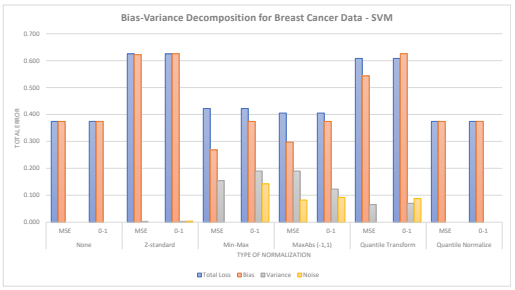
(A) Logistic Regression



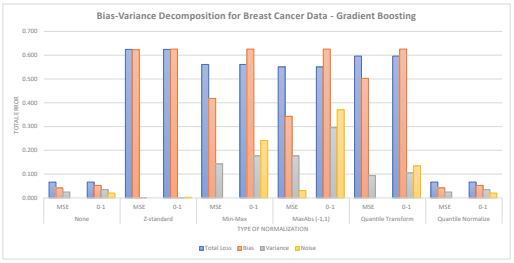
(B) Decision Tree



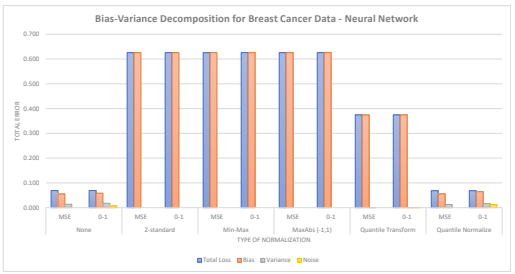
(C) Random Forest



(D) Support Vector Machine (SVM)



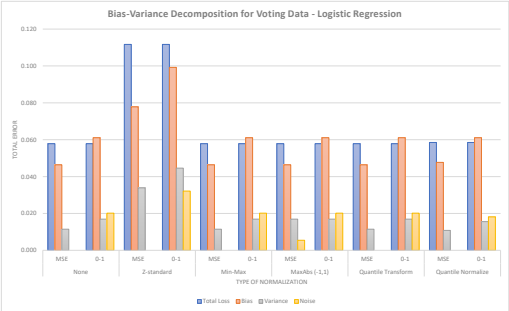
(E) Gradient Boosting Regression (GBR)



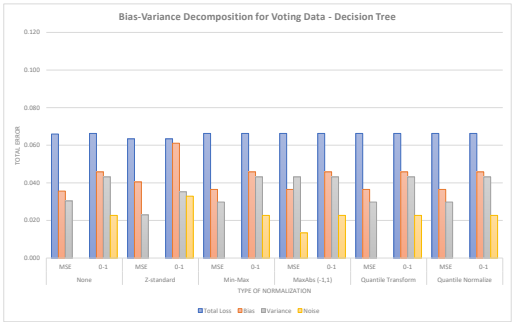
(F) Neural Network

FIGURE B.14: Bias-Variance Decomposition for Breast Cancer Data with Binary Target

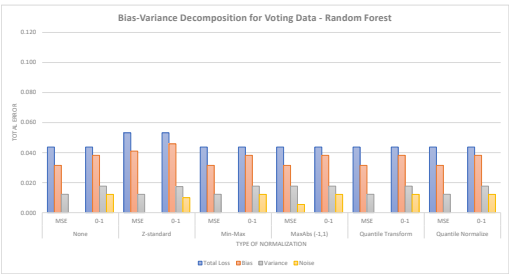
Voting Data



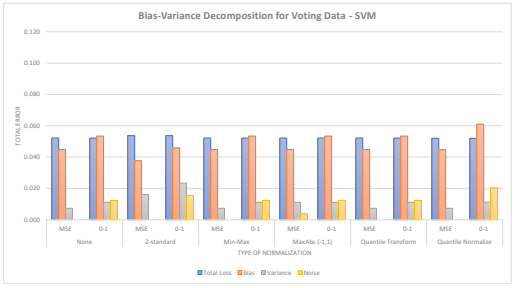
(A) Logistic Regression



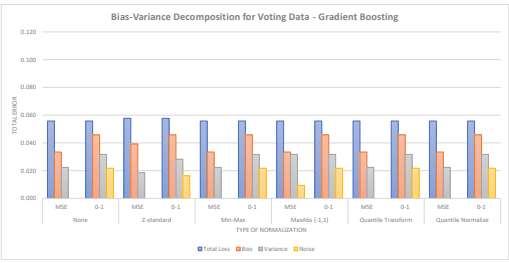
(B) Decision Tree



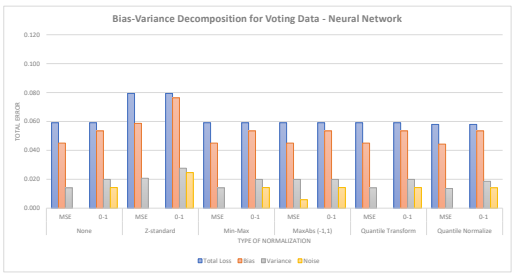
(C) Random Forest



(D) Support Vector Machine (SVM)



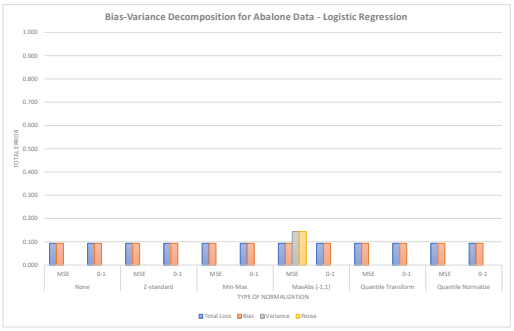
(E) Gradient Boosting Regression (GBR)



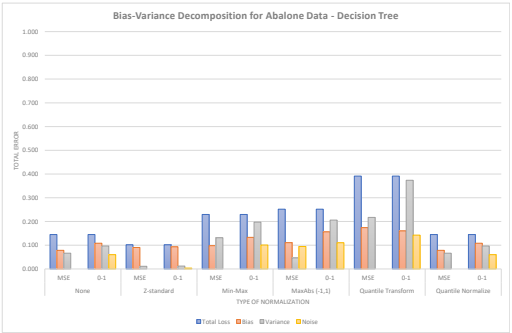
(F) Neural Network

FIGURE B.15: Bias-Variance Decomposition for Congressional Voting Data with Binary Target

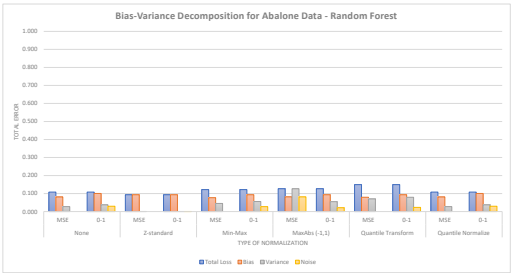
Abalone Data



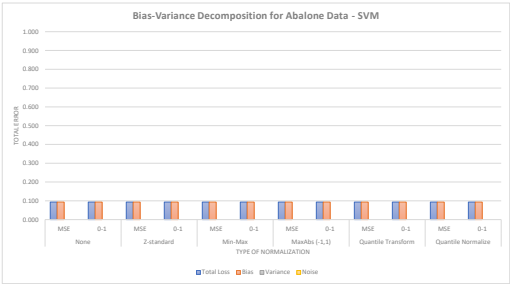
(A) Logistic Regression



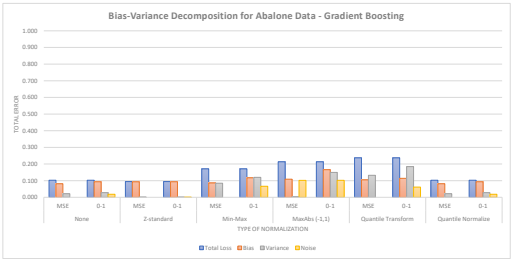
(B) Decision Tree



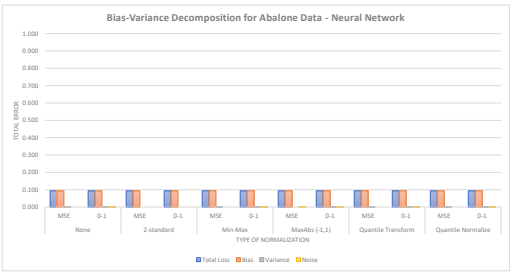
(C) Random Forest



(D) Support Vector Machine (SVM)



(E) Gradient Boosting Regression (GBR)

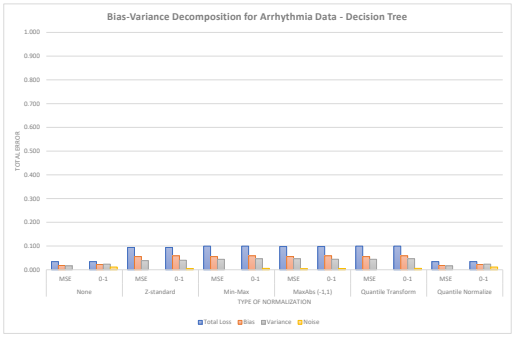


(F) Neural Network

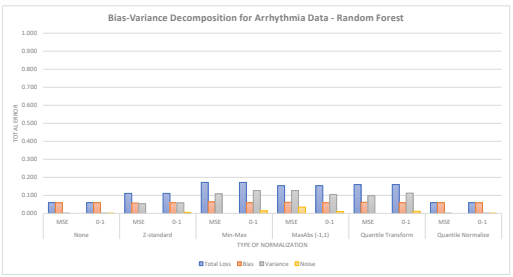
FIGURE B.16: Bias-Variance Decomposition for Abalone Data with Binary Target

Arrhythmia Data

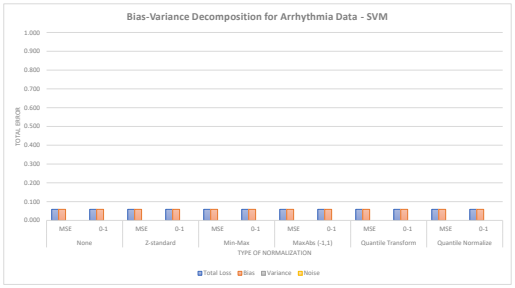
(A) Logistic Regression



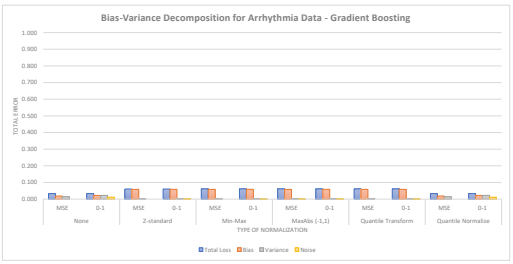
(B) Decision Tree



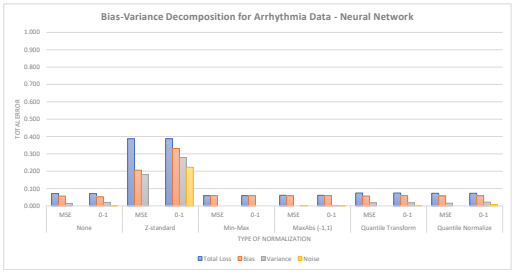
(C) Random Forest



(D) Support Vector Machine (SVM)



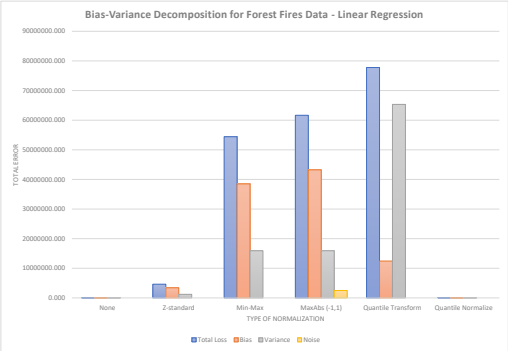
(E) Gradient Boosting Regression (GBR)



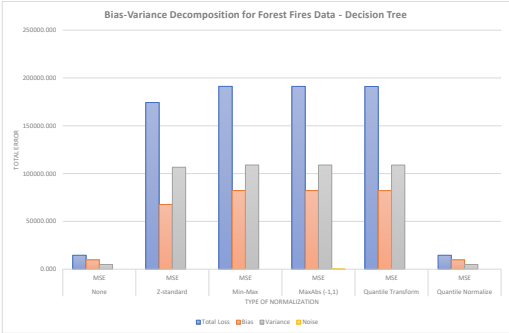
(F) Neural Network

FIGURE B.17: Bias-Variance Decomposition for Arrhythmia Data with Binary Target

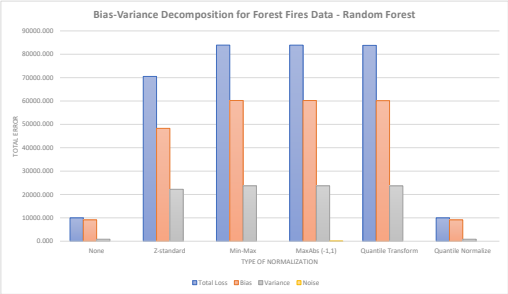
Forest Fires Data



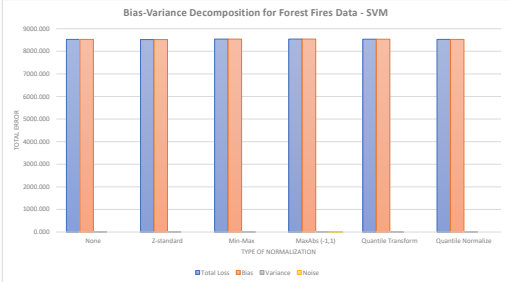
(A) Linear Regression



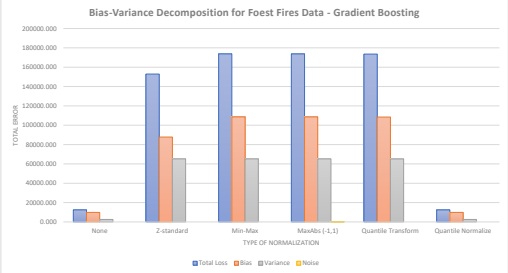
(B) Decision Tree



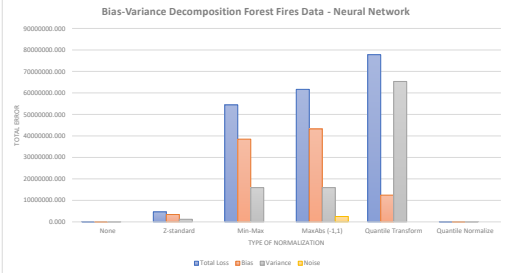
(C) Random Forest



(D) Support Vector Machine (SVM)



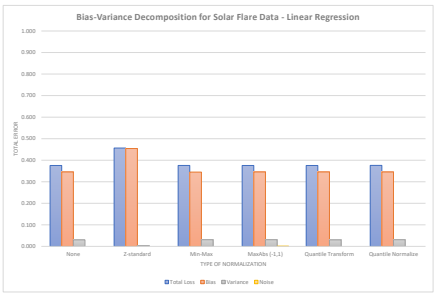
(E) Gradient Boosting Regression (GBR)



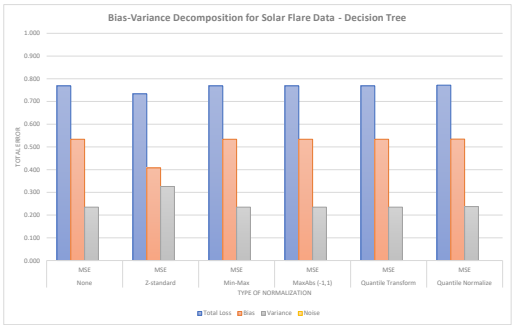
(F) Neural Network

FIGURE B.18: Bias-Variance Decomposition for Forest Fires Data with Continuous Target

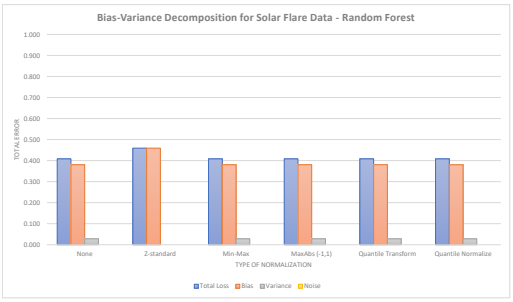
Solar Flares Data



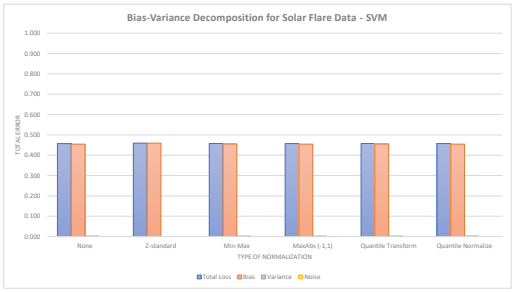
(A) Linear Regression



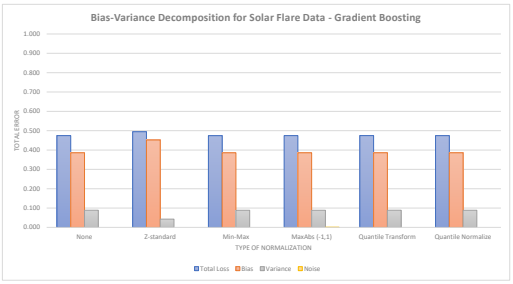
(B) Decision Tree



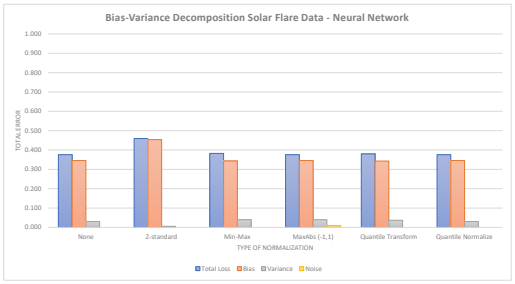
(C) Random Forest



(D) Support Vector Machine (SVM)



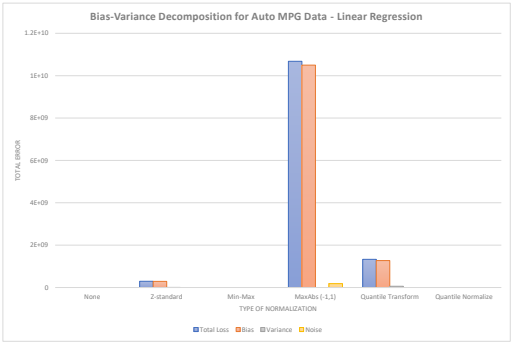
(E) Gradient Boosting Regression (GBR)



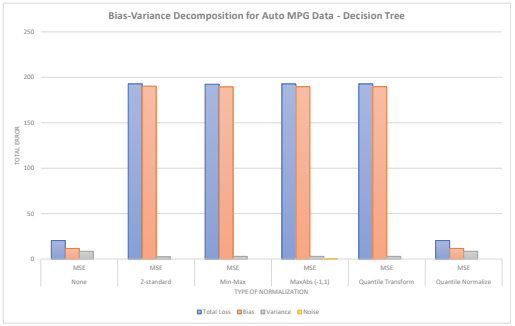
(F) Neural Network

FIGURE B.19: Bias-Variance Decomposition for Solar Flares Data with Continuous Target

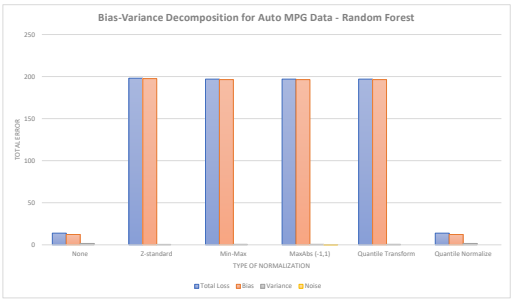
Auto MPG Data



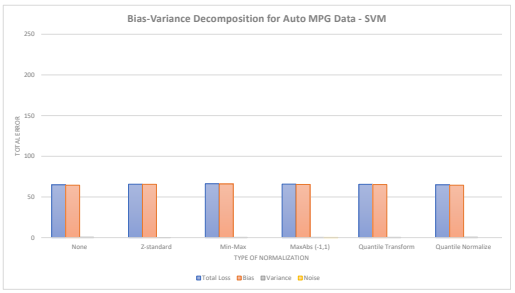
(A) Linear Regression



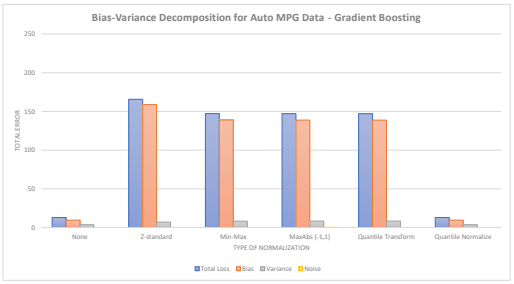
(B) Decision Tree



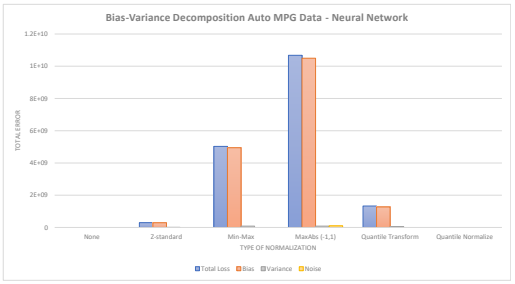
(C) Random Forest



(D) Support Vector Machine (SVM)



(E) Gradient Boosting Regression (GBR)



(F) Neural Network

FIGURE B.20: Bias-Variance Decomposition for Auto MPG Data with Continuous Target

Appendix C

Code

```
## ---------------------------
##
## Script name: CreateSimData.R
##
## Purpose of script: This script simulates 11 types of datasets for
dissertation simulation study:
## bivariate normal (continuous and binary outcome), rank-based (continuous
and binary outcome),
## categorical (continuous and binary outcome), mixed data (continuous and
binary outcome),
## poisson (bivariate normal, categorical, ranked, mixed data with poisson
response)
##
## Output of script: 9 simulated datasets
##
## Author: Jessica M. Rudd, MPH
##
## Date Created: 2020-03-16
##
## Copyright (c) Jessica M. Rudd 2020
## Email: jess@irudd.com
##
## ---------------------------
##
## Notes:
##
##
## ---------------------------

## set working directory for Mac and PC

setwd("C:/Users/jess/OneDrive/Grad School/Dissertation/Programs/Simulations")
# Tim's working directory (PC)

## ---------------------------

options(scipen = 6, digits = 4) # I prefer to view outputs in non-scientific
notation
#memory.limit(30000000)     # this is needed on some PCs to increase memory
allowance, but has no impact on macs.

## load up the packages we will need:  (uncomment as required)
library(dplyr)
library(MASS)
library(arm)
library(Hmisc)
library(dummies)

# number of simulated observations
n <- 1000

# Target parameters for univariate normal distributions
rho <- 0.0
mu1 <- 10
```

```
s1 <- 1
mu2 <- 150
s2 <- 2

#Linear coefficients
b0 <- 1
b1 <- 1.5
b2 <- 2

#Simulate bivariate normal data with binary outcome
set.seed(123)

sim_bvn_bin <- function() {
  # Create dependent variables
  x1 <- mvrnorm(n, mu1, s1)
  x2 <- mvrnorm(n, mu2, s2)
  eps = rnorm(n = n, mean = 0, sd = .1) #irreducible error
  pr = invlogit((b0 + b1*x1 + b2*x2 + eps)*.001)
  Y = rbinom(n, 1, pr/2)

  bivarNorm_bin <- data.frame(Y = Y, x1 = x1, x2 = x2)
  #Save as CSV
  write.csv(bivarNorm_bin,'bivarNorm_bin.csv')
  return(bivarNorm_bin)
}
```

Bias-Variance Decomposition Sample

May 10, 2020

[ ]:
```
#from imblearn.datasets import fetch_datasets
from sklearn import preprocessing
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import itertools
#from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn import svm
from sklearn.svm import SVC
#from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
#from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
#from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import BaggingRegressor
from sklearn.neural_network import MLPRegressor
import csv
import time
import os
import logging

import json
from datapackage import Package

#package = Package('https://datahub.io/machine-learning/abalone/datapackage.
 ↪json')
#package = Package('https://datahub.io/machine-learning/arrhythmia/datapackage.
 ↪json')
```

[ ]:
```
logging.basicConfig(filename="mse_decomp_standard_bivarNormCont.log",␣
 ↪level=logging.DEBUG)
```

```
[ ]: bivarNorm_cont = pd.read_csv('C:/Users/jess/OneDrive/Grad School/Dissertation/
     ↪Programs/Simulations/bivarNorm_cont.csv')
```

```
[ ]: bivarNorm_cont.head()
```

```
[ ]: bivarNorm_cont.describe()
```

0.0.1 Bias-Variance decomposition adapted from mlxtend package created by Sebastian Raschka (https://github.com/rasbt)

```
[ ]: def _draw_bootstrap_sample(rng, X, y):
         sample_indices = np.arange(X.shape[0])
         bootstrap_indices = rng.choice(sample_indices,
                                        size=sample_indices.shape[0],
                                        replace=True)
         return X[bootstrap_indices], y[bootstrap_indices]
```

```
[ ]: def bias_variance_decomp_mse(estimator, X_train, y_train, X_test, y_test,
                                 loss='0-1_loss', num_rounds=200, random_seed=None):

         supported = ['0-1_loss', 'mse']

         if loss not in supported:
             raise NotImplementedError('loss must be one of the following: %s' %
                                       supported)

         rng = np.random.RandomState(random_seed)

         all_pred = np.zeros((num_rounds, y_test.shape[0]), dtype=np.int)
         #all_pred_df = pd.DataFrame()

         for i in range(num_rounds):
             X_boot, y_boot = _draw_bootstrap_sample(rng, X_train, y_train)
             #estimator.fit(X_boot, y_boot)
             pred = estimator.fit(X_boot, y_boot).predict(X_test)
             #all_pred_df[i] = pred
             all_pred[i] = pred

         #all_pred = all_pred_df.values.T

         if loss == '0-1_loss':
             main_predictions = np.apply_along_axis(lambda x:
                                                    np.argmax(np.bincount(x)),
                                                    axis=0,
                                                    arr=all_pred)


             avg_expected_loss = (main_predictions != y_test).sum()/y_test.size
```

```
        avg_expected_loss = np.apply_along_axis(lambda x:
                                                (x != y_test).mean(),
                                                axis=1,
                                                arr=all_pred).mean()

        avg_bias = np.sum(main_predictions != y_test) / y_test.size

        var = np.zeros(pred.shape)

        for pred in all_pred:
            var += (pred != main_predictions).astype(np.int)
        var /= num_rounds

        avg_var = var.sum()/y_test.shape[0]

    else:
        avg_expected_loss = np.apply_along_axis(
            lambda x:
            ((x - y_test)**2).mean(),
            axis=1,
            arr=all_pred).mean()

        main_predictions = np.mean(all_pred, axis=0)

        avg_bias = np.sum((main_predictions - y_test)**2) / y_test.size
        avg_var = np.sum((main_predictions - all_pred)**2) / all_pred.size

        #all_pred_df["true"] = y_test

    return avg_expected_loss, avg_bias, avg_var #, all_pred_df
```

[ ]:
```
### Bivariate Normal Linear with raw data, MSE
dataset_names = [bivarNorm_cont]
name = 'bivarNorm_cont'
loss = 'mse'
normalization ='None'
model_name = 'Linear Regression'

bootstrap_rounds = 1000

for dataset_name in dataset_names:

    logging.info("\ndataset: %s" % dataset_name)
```

```
    result_csv = [f for f in os.listdir("C:/Users/jess/OneDrive/Grad School/
↪Dissertation/Programs/mse_decomp_results") if name+"_bootstrap_standard.csv"␣
↪in f]
    if len(result_csv) == 1:
        logging.info("This dataset is done!")

    else:
        dataset = dataset_name
        dataset.data = dataset.iloc[:,2:].values
        dataset.y = dataset.iloc[:,1].values


        X_train, X_test, y_train, y_test = train_test_split(dataset.data,␣
↪dataset.y,
                                                            test_size=0.3,␣
↪random_state=0)

        model = LinearRegression()
        #classifier = DecisionTreeRegressor(random_state=123)

        time1 = time.time()


        avg_expected_loss, avg_bias, avg_var = bias_variance_decomp_mse(
                model, X_train, y_train, X_test, y_test, loss='mse',␣
↪num_rounds=bootstrap_rounds, random_seed=123)


        time2 = time.time()
        standard_time = time2-time1
        logging.info("running time of standard model in seconds: %.2f" %␣
↪standard_time)
        logging.info("average expected loss: %s" % avg_expected_loss)
        logging.info("average bias: %s " % avg_bias)
        logging.info("average variance: %s " % avg_var)
```

[ ]:
```
# Store model results
summary1 = pd.DataFrame({'Dataset': [name], 'Normalization': [normalization],␣
↪'Loss Function': [loss], 'Model': [model_name],
                                'avg_expected_loss': [avg_expected_loss], 'Avg.␣
↪Bias': [avg_bias], 'Avg. Variance': [avg_var]},
                               columns=['Dataset', 'Normalization', 'Loss␣
↪Function', 'Model', 'avg_expected_loss', 'Avg. Bias', 'Avg. Variance'])
```

Bibliography

Agresti, A. (2003). *Categorical data analysis*. John Wiley & Sons.

Altman, M., Gill, J., & McDonald, M. P. (2004). *Numerical issues in statistical computing for the social scientist*. John Wiley & Sons.

Applying Machine Learning To March Madness. (n.d.). Retrieved from https://www.kdnuggets.com/2017/03/machine-learning-march-madness.html

Argyropoulos, C., Chatziioannou, A., Nikiforidis, G., Moustakas, A., Kollias, G., & Aidinis, V. (2006). Operational criteria for selecting a cdna microarray data normalization algorithm. *Oncology reports*, *15*(4), 983–996.

Asmann, Y. W., Necela, B. M., Kalari, K. R., Hossain, A., Baker, T. R., Carr, J. M., ... Li, X. [Xing], et al. (2012). Detection of redundant fusion transcripts as biomarkers or disease-specific therapeutic targets in breast cancer. *Cancer research*, *72*(8), 1921–1928.

Barakat, M. S., Field, M., Ghose, A., Stirling, D., Holloway, L., Vinod, S., ... Thwaites, D. (2017). The effect of imputing missing clinical attribute values on training lung cancer survival prediction model performance. *Health information science and systems*, *5*(1), 16.

Botvinik-Nezer, R., Holzmeister, F., Camerer, C. F., Dreber, A., Huber, J., Johannesson, M., ... Adcock, R. A., et al. (2020). Variability in the analysis of a single neuroimaging dataset by many teams. *Nature*, 1–7.

Brown, M., Kvam, P., Nemhauser, G., & Sokol, J. (2012, March). Insights from the lrmc method for ncaa tournament predictions. mit sloan sports conference.

Brownlee, J. (2019). How to use data scaling improve deep learning model stability and performance. Retrieved from https://machinelearningmastery.com/how-to-improve-neural-network-%20stability-and-modeling-performance-with-data-scaling/

Bryan, K., Steinke, M., & Wilkins, N. (2006). Upset special: Are march madness upsets predictable? *Available at SSRN 899702*.

Bullard, J., Purdom, E., Hansen, K., & Dudoit, S. (2010). Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments. *BMC Bioinformatics*, *11*(1), 94.

Cai, S., Gao, J., Zhang, M., Wang, W., Chen, G., & Ooi, B. C. (2019). Effective and efficient dropout for deep convolutional neural networks. *arXiv preprint arXiv:1904.03392*.

Carp, J. (2012a). On the plurality of (methodological) worlds: Estimating the analytic flexibility of fmri experiments. *Frontiers in neuroscience*, *6*, 149.

Carp, J. (2012b). The secret lives of experiments: Methods reporting in the fmri literature. *Neuroimage*, *63*(1), 289–300.

Chakure, A. (2020). Random forest and its implementation. Towards Data Science. Retrieved from https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

Cheung, V. G., Nayak, R. R., Wang, I. X., Elwyn, S., Cousins, S. M., Morley, M., & Spielman, R. S. (2010). Polymorphic cis-and trans-regulation of human gene expression. *PLoS biology*, *8*(9), e1000480.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

Davenport, T., & Harris, J. (2017). *Competing on analytics: Updated, with a new introduction: The new science of winning*. Harvard Business Press.

Dietterich, T. G., & Kong, E. B. (1995). *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms*. Technical report, Department of Computer Science, Oregon State University.

Dinga, R., Penninx, B. W., Veltman, D. J., Schmaal, L., & Marquand, A. F. (2019). Beyond accuracy: Measures for assessing machine learning models, pitfalls and guidelines. *bioRxiv*, 743138.

Domingos, P. (2000). A unified bias-variance decomposition. In *Proceedings of 17th international conference on machine learning* (pp. 231–238).

Dua, D., & Graff, C. (2017). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. Retrieved from http://archive.ics.uci.edu/ml

Dutta, S., Jacobson, S., & Sauppe, J. (2017). Identifying ncaa tournament upsets using balance optimization subset selection. *Journal of Quantitative Analysis in Sports*, *13*(2), 79–93.

Evans, C., Hardin, J., & Stoebel, D. (2017). Selecting between-sample rna-seq normalization methods from the perspective of their assumptions. *Briefings in bioinformatics*, *19*(5), 776–792.

Floridi, L. (2012). Big data and their epistemological challenge. *Philosophy & Technology*, *25*(4), 435–437.

Ford, K., & Fodor, A. (2018). Predicting upsets: The 2017 ncaa men's basketball tournament. *Journal of Prediction Markets*, *12*(1).

Forman, G., Scholz, M., & Rajaram, S. (2009). Feature shaping for linear svm classifiers. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 299–308).

Franc, V., Zien, A., & Schölkopf, B. (2011). Support vector machines as probabilistic models. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 665–672).

Google Cloud NCAA ML Competition. (2020). Retrieved from https://www.kaggle.com/c/google-cloud-ncaa-march-madness-2020-division-1-mens-tournament/overview

Gravlee, C. C., Bernard, H. R., & Leonard, W. R. (2003). Heredity, environment, and cranial form: A reanalysis of boas's immigrant data. *American Anthropologist*, *105*(1), 125–138.

Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2011). The misuse of the nasa metrics data program data sets for automated software defect prediction. In *15th annual conference on evaluation & assessment in software engineering (ease 2011)* (pp. 96–103). IET.

Grover, P. (2019). Gradient boosting from scratch. Retrieved from https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d

Hao, Y., Kristal, B., & Hsu, D. (2018). Predication of ncaa bracket using recurrent neural network and combinatorial fusion. In *2018 ieee 16th intl conf on dependable, autonomic and secure computing, 16th intl conf on pervasive intelligence and computing, 4th intl conf on big data intelligence and computing and cyber science and technology congress (dasc/picom/datacom/cyberscitech)* (pp. 897–903).

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 558–567).

Hotelling, H. (1940). The teaching of statistics. *The Annals of Mathematical Statistics*, *11*(4), 457–470.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70*(1-3), 489–501.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jordan, J. (2019). Normalizing your data (specifically, input and batch normalization). Jeremy Jordan. Retrieved from https://www.jeremyjordan.me/batch-normalization/

Kaplan, E. H., & Garstka, S. J. (2001). March madness and the office pool. *Management Science*, *47*(3), 369–382.

Kim, S.-Y. (2009). Effects of sample size on robustness and prediction accuracy of a prognostic gene signature. *BMC bioinformatics*, *10*(1), 147.

Kitchin, R. (2014). Big data, new epistemologies & paradigm shifts. *Big data and society*, *1*(1).

Kohavi, R., Wolpert, D. H. et al. (1996). Bias plus variance decomposition for zero-one loss functions. In *Icml* (Vol. 96, pp. 275–83).

Kondo, M., Bezemer, C.-P., Kamei, Y., Hassan, A. E., & Mizuno, O. (2019). The impact of feature reduction techniques on defect prediction models. *Empirical Software Engineering*, 1–39.

Kumari, V. A., & Chitra, R. (2013). Classification of diabetes disease using support vector machine. *International Journal of Engineering Research and Applications*, *3*(2), 1797–1801.

Kvam, P., & Sokol, J. S. (2006). A logistic regression/markov chain model for ncaa basketball. *Naval Research Logistics (NrL)*, *53*(8), 788–803.

Lakshmanan. (2019). How, when and why should you normalize/standardize/rescale your data? Medium. Retrieved from https://medium.com/@swethalakshmanan14/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

Leonelli, S. (2014). What difference does quantity make? on the epistemology of big data in biology. *Big data & society*, *1*(1), 2053951714534395.

Li, X. [Xiang], Chen, S., Hu, X., & Yang, J. (2019). Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2682–2690).

Littler, S. (2018). The importance and effect of sample size. Retrieved from https://select-statistics.co.uk/blog/importance-effect-sample-size/

Lowrie, I. (2017). Algorithmic rationality: Epistemology and efficiency in the data sciences. *Big Data & Society*, *4*(1), 2053951717700925.

Mayer-Schönberger, V., & Cukier, K. (2013). *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt.

Ng, A. (2020). Orthogonalization - ml strategy. Retrieved from https://www.coursera.org/learn/machine-learning-projects/lecture/FRvQe/orthogonalization

Normalization. (n.d.). Retrieved from https://www.codecademy.com/articles/normalization

Organizing Your Social Sciences Research Paper: Theoretical Framework. (2020). Retrieved from https://libguides.usc.edu/writingguide/theoreticalframework

Owen, S., Ryza, S., Laserson, U., & Wills, J. (2015). Advanced analytics with apache spark. O'Reilly Media.

Rai, S., Ray, H., Yuan, X., Pan, J., Hamid, T., & Prabhu, S. (2012). Statistical analysis of repeated microrna high-throughput data with application to human heart failure: A review of methodology. *Open access medical statistics*, *2012*(2), 21.

Raschka, S. (n.d.). Bias-variance decomposition. Retrieved from http://rasbt.github.io/mlxtend/user_guide/evaluate/bias_variance_decomp/

Robinson, M., & Oshlack, A. (2010). A scaling normalization method for differential expression analysis of rna-seq data. *Genome Biology*, *11*(3), 25.

Rudd, J. M. (2018). Application of support vector machine modeling and graph theory metrics for disease classification. *Model Assisted Statistics and Applications*, *13*(4), 341–349.

Rudd, J. M., & Priestley, J. L. (2017). A comparison of decision tree with logistic regression model for prediction of worst non-financial payment status in commercial credit.

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? In *Advances in neural information processing systems* (pp. 2483–2493).

Schwartz, N. (2015). Duke math professor says odds of a perfect bracket are one in 2.4 trillion. Gannett Satellite Information Network. Retrieved

from https://ftw.usatoday.com/2015/03/duke-math-professor-says-odds-of-a-perfect-bracket-are-one-in-2-4-trillion

Schwarzenbach, H., Silva, A. D., Calin, G., & Pantel, K. (2015). Data normalization strategies for microrna quantification. *Clinical chemistry*, *61*(11), 1333–1342.

Shah, T. (2017). About train, validation and test sets in machine learning. Towards Data Science. Retrieved from https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

Shi, Z., Moorthy, S., & Zimmermann, A. (2013, September). Predicting ncaab match outcomes using ml techniques-some results and lessons learned. in proceedings of the mlsa workshop at ecml/pkdd.

Shulga, D. (2018). 5 reasons why you should use cross-validation in your data science projects. Towards Data Science. Retrieved from https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79

Silberzahn, R., Uhlmann, E. L., Martin, D. P., Anselmi, P., Aust, F., Awtrey, E., ... Bonnier, E., et al. (2018). Many analysts, one data set: Making transparent how variations in analytic choices affect results. *Advances in Methods and Practices in Psychological Science*, *1*(3), 337–356.

Singh, S. (2018). Understanding the bias-variance tradeoff. Towards Data Science. Retrieved from https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Song, Z., Guo, Y., Wu, Y., & Ma, J. (2019). Short-term traffic speed prediction under different data collection time intervals using a sarima-sdgm hybrid prediction model. *PloS one*, *14*(6), e0218626.

Tukey, J. W. (1962). The future of data analysis. *The annals of mathematical statistics*, *33*(1), 1–67.

Tukey, J. W. (1977). *Exploratory data analysis*. Reading, Mass.

Vincent, J. (2018). Ibm hopes to fight bias in facial recognition with new diverse dataset. The Verge. Retrieved from https://www.theverge.com/2018/6/27/17509400/facial-recognition-bias-ibm-data-training

Vinz, S. (2019). The theoretical framework: What and how?: A quick guide. Retrieved from https://www.scribbr.com/dissertation/theoretical-framework/

Wagenmakers, E.-J., Wetzels, R., Borsboom, D., van der Maas, H. L., & Kievit, R. A. (2012). An agenda for purely confirmatory research. *Perspectives on Psychological Science*, *7*(6), 632–638.

What is Data Cleansing?: Experian Business. (n.d.). Retrieved from https://www.experian.co.uk/business/glossary/data-cleansing/

Wolpert, D. H., Macready, W. G. et al. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67–82.

Xu, Y., & Goodacre, R. (2018). On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*, *2*(3), 249–262.

Yuan, L., Liu, A., Yeh, A., Kaufman, A., Reece, A., Bull, P., & Bornn, L. (2015). A mixture-of-modelers approach to forecasting ncaa tournament outcomes. *Journal of Quantitative Analysis in Sports*, *11*(1), 13–27.

Zeng, N. (2018). Understanding the bias-variance decomposition with a simulated experiment: Nickzeng. Retrieved from https://blog.zenggyu.com/en/post/2018-03-11/understanding-the-bias-variance-decomposition-with-a-simulated-experiment/

Zhang, L., Ray, H., Priestley, J., & Tan, S. (2019). A descriptive study of variable discretization and cost-sensitive logistic regression on imbalanced credit data. *Journal of Applied Statistics*, 1–14.

Zyprych-Walczak, J., Szabelska, A., Handschuh, L., Górczak, K., Klamecka, K., Figlerowicz, M., & Siatkowski, I. (2015). The impact of normalization methods on rna-seq data analysis. *BioMed research international*, *2015*.