UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Robust Interactions with Machine Learning Models

Permalink

https://escholarship.org/uc/item/1gb45688

Author

Slack, Dylan

Publication Date

2023

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, availalbe at https://creativecommons.org/licenses/by/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org
Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

Robust Interactions with Machine Learning Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Dylan Slack

Dissertation Committee:
Associate Professor Sameer Singh, Chair
Assistant Professor Himabindu Lakkaraju
Professor Erik Sudderth
Chancellor's Professor Padhraic Smyth

2023

© 2023 Dylan Slack

DEDICATION

To my family.

TABLE OF CONTENTS

Page

LIST OF FIGURES **vi**

LIST OF TABLES **x**

LIST OF ALGORITHMS **xii**

ACKNOWLEDGMENTS **xiii**

VITA **xv**

ABSTRACT OF THE DISSERTATION **xviii**

1 Introduction **1**
- 1.1 Contributions & Key Findings 3
  - 1.1.1 Robust Explainability 3
  - 1.1.2 Conversational Systems for ML Development 5
- 1.2 Declaration of Previous Work and Collaborations 6

I Robust Machine Learning Explanations **8**

2 Explainability Background **9**
- 2.1 Notation 10
- 2.2 Local Model Agnostic Explanations 10
- 2.3 Counterfactual Explanations 12

3 Adversarial Attacks On Local Post Hoc Explanations **16**
- 3.1 Proposed Framework 17
- 3.2 Experimental Results 20
  - 3.2.1 Effectiveness of Adversarial Classifiers 23
  - 3.2.2 Effect of Perturbation Detection Accuracy 26
  - 3.2.3 Robustness to Hyperparameters 27
  - 3.2.4 Example Explanations 28
- 3.3 Summary of Contributions 28

4 Adversarial Attacks on Counterfactual Explanations 30
4.1 Adversarial Models for Manipulating Counterfactual Explanations 33
4.2 Experiment Setup 38
4.3 Experiments 40
4.3.1 Effectiveness of the Manipulation 41
4.3.2 Outlier Factor of Counterfactuals 42
4.3.3 Potential Mitigation Strategies 43
4.4 Potential Impacts 45
4.5 Discussion 46
4.6 Summary of Contribution 47

5 Reliable Local Post Hoc Explanations 48
5.1 Our Framework: Bayesian Local Explanations 50
5.1.1 Constructing Bayesian Local Explanations 51
5.1.2 Estimating the Number of Perturbations 55
5.1.3 Focused Sampling of Perturbations 57
5.2 Experiments 59
5.3 Discussion 65
5.4 Summary of Contributions 65

II Natural Language For Explaining & Developing Machine Learning Models 67

6 Natural Language Conversations For Explainability With TalkToModel 68
6.1 User Study 69
6.1.1 Format 70
6.1.2 Results 71
6.1.3 Explainability Dialogue Desiderata from Interviewees 73
6.2 TalkToModel 74
6.3 Methods 77
6.3.1 Text Understanding 77
6.3.2 Executing Parses 83
6.4 Results 88
6.4.1 Advantages of Explanation Selection 92
6.5 Summary of Contribution 99

7 TABLET: Natural Language Instructions for Tabular Prediction 101
7.1 Tabular Instruction Learning 104
7.1.1 Problem Formulation 104
7.1.2 Prompting Schema 105
7.2 TABLET 106
7.2.1 Tasks 106
7.2.2 Instructions 107
7.3 Experimental Setup 109

7.4 Experiments . . . 111
    7.4.1 Zero-Shot Performance . . . 112
    7.4.2 Few-Shot Performance . . . 114
7.5 Summary Of Contribution . . . 117

8 Conclusions and Future Work 119
8.1 Impact . . . 120
8.2 Future Work . . . 121

Bibliography 123

Appendix A Reliable Post hoc Explanations 136

Appendix B TalkToModel 141

LIST OF FIGURES

Page

2.1 Example LIME model agnostic explanations on a tabular and image classification problem, demonstrating local explanations for different data modalities and classes. Green indicates *positive* contribution for the model predicting the class, while red indicates *negative*. The leftmost graph (a) includes an explanation for a `benign` prediction for a random forest training to predict the presence of breast cancer. The center (b) and right (c) images are explanations for the `St. Bernard` and `Pyrenees` classes for the Inception V3 neural network. 11

2.2 We compare two choices of distance functions on the search for counterfactual explanations (CFEs) on a toy problem. The start of the search for the CFE is the point in blue within the negative classification region. The Wachter et. al's distance function (green) emphasizes closeness to the search initialization and the CFE being classified positively. The distance function from Van Looveren and Klaise [2019] (purple) additionally emphasizes *distributionality* of the CFE, meaning it should be more realistic for achieving. While both these algorithms find valid CFEs, they are quite different because the objectives emphasize different desirable properties of CFEs. 14

3.1 PCA applied to the COMPAS dataset (blue) as well as its LIME style perturbations (red). Even in this low-dimensional space, we can see that data points generated via perturbations are distributed very differently from instances in the COMPAS data. In this paper, we exploit this difference to craft adversarial classifiers. 17

3.2 **COMPAS:** % of data points for which each feature (color coded) shows up in top 3 (according to LIME and SHAP's ranking of feature importances) for the biased classifier f (left), our adversarial classifier where ψ uses only one uncorrelated feature to make predictions (middle), and our adversarial classifier where ψ uses two uncorrelated features to make predictions (right). 23

3.3 **Communities and Crime:** Similar to Fig 3.2; *Race % White* is the sensitive feature here. 24

3.4 **German credit:** Similar to Fig 3.2 and 3.3, but unbiased classifier ψ uses an existing feature (*Loan Rate % Income*) to make predictions, and *Gender* is the sensitive feature. Feature importances for the biased classifier f shown in the figure (left) are generated using LIME; SHAP also produces similar feature importance values. 25

3.5 Effectiveness of our attack: % of data points for which race is the most important feature vs. the F1 score of the OOD classifier (perturbation detection) for both LIME and SHAP on the COMPAS dataset over a variety of models (fit with line plot over mean of binned points). 26
3.6 Example SHAP explanations for a data point with biased classifier f (top) and adversarial classifier e (bot.). 28

4.1 **Model trained with BCE objective and adversarial model on a toy data set** using Wachter et al.'s Algorithm [Wachter et al., 2018]. The surface shown is the loss in Wachter et al.'s Algorithm with respect to $\boldsymbol{x}$, the line is the path of the counterfactual search, and we show results for a single point, $\boldsymbol{x}$. For the model without the manipulation (subfigure 4.1a), the counterfactuals for $\boldsymbol{x}$ and $\boldsymbol{x} + \boldsymbol{\delta}$ converge to the same minima and are similiar cost recourse. For the adversarial model (subfigure 4.1b), the recourse found for $\boldsymbol{x}$ has *higher cost* than $\boldsymbol{x} + \boldsymbol{\delta}$ because the local minimum initialized at $\boldsymbol{x}$ is *farther* than the minimum starting at $\boldsymbol{x} + \boldsymbol{\delta}$, demonstrating the problematic behavior of counterfactual explanations. 31
4.2 **Manipulated Model for Loan Risk.** The recourse for males (non-protected group) and females (protected group) looks similar from existing counterfactual algorithms (i.e. model seems fair). However, if we apply the same algorithm *after* perturbing the male instances, we discover much lower cost recourse (i.e. the model discriminates between sexes). 32
4.3 **Outlier Factor of Counterfactuals:** For the Wachter et al.'s and DiCE models for Communities and Crime, we show that the manipulated recourses are only slightly less realistic than counterfactuals of the unmodified model, whereas the counterfactuals found after adding $\boldsymbol{\delta}$ are more realistic than the original counterfactuals (lower is better). 42
4.4 **Exploring Mitigation Strategies:** For the Wachter et al. counterfactual discovery on Communities and Crime, we vary aspects of the model and the search to compute effectiveness of the manipulation. Each provides a potentially viable defense, with different trade-offs. 43

5.1 **Example explanations** on for an instance from the COMPAS dataset, where vertical lines indicate the feature importance by LIME (red is negative effect, green is positive) and the shaded region visualizes the uncertainty estimated by BayesLIME. While LIME produces very different and contradictory feature importance for different number of perturbations (5.1a and 5.1b), BayesLIME provides more context. The overlapping uncertainty intervals in the explanation computed with 100 perturbations (5.1a) indicate that it is unclear which feature is the most important. However, the tighter uncertainty intervals in the explanation computed with 2K perturbations (5.1b) clearly indicates that `Female` is the most important. 49

5.2 Rerunning LIME local explanations 1000 times and BayesLIME *once* for linear and non-linear toy surfaces using few (25) and many (250) perturbations. The linear surface is given as $p(y) \propto x_1$ and the non linear surface is defined as $p(y) \propto \sin(x_1/2)*10+\cos(10+(x_1*x_2)/2)*\cos(x_1)$. We plot each run of LIME in blue and the BayesLIME 95% credible region of the feature importance ϕ in black. We see that LIME variance is higher with fewer samples and a less linear surface. BayesLIME captures the relative difficulty of explaining each surface through the width the credible region. For instance, BayesLIME is most uncertain in the nonlinear, few samples case because this surface is the most difficult to explain. 54
5.3 **Perturbations-to-go (G).** We generate explanation with G perturbations, where G is computed using the *desired* credible interval width (x-axis), and compare desired levels to the *observed* credible interval width (y-axis) (blue line indicates ideal calibration). Results are averaged over 100 MNIST images of the digit "4" We see that G provides a good approximation of the additional perturbations needed. 61
5.4 **Efficiency of focused sampling** for 100 Imagenet "French bulldog" images, with random sampling as a baseline. We provide mean and standard error. We assess the efficiency of focused sampling by comparing *error uncertainty* over model queries and show quicker convergence than random sampling. . . 63
5.5 **Assessing the % increase in stability** of BayesLIME and BayesSHAP over LIME and SHAP respectively. Our Bayesian methods are significant more stable ($\rho <$ 1e-2 according to Wilcoxon signed-rank test) except for BayesSHAP on German Credit, where there is not a significant difference between the methods ($\rho > 0.05$). 63

6.1 **Overview of TalkToModel:** Instead of writing code, users have conversations with TalkToModel as follows: **(1)** users supply natural language inputs. **(2)** the dialogue engine parses the input into an executable representation. **(3)** the execution engine runs the operations and the dialogue engine uses the results in its response. 76

7.1 **Evaluation with TABLET.** We serialize a short task description and a data point from the Whooping Cough dataset into a prompt (top). Flan-T5 11b and ChatGPT predict this instance incorrectly based on clues related to a cough and nasal congestion. However, by adding the instructions (bottom), the model identifies that the patient's pain is inconsistent with typical symptoms of Whooping Cough and correctly predicts the instance. 102
7.2 **Instruction generation pipeline.** (1) We fit a prototypes and rulesets classifier on the dataset; here Adult (2) We serialize the classifier's logic into a template (3) We sample revisions of the template with GPT-3. 107
7.3 **Results on DDX tasks with Naturally Occurring Instructions.** Instructions greatly improve LLM generalization of ChatGPT, Flan-T5 11b, and Tk-Instruct 11b over the baseline without instructions (LIFT). 109

7.4 **Model results on all tasks with generated instructions.** Both prototypes and rulesets generated instructions lead to improved performance on most tasks for Flan-T5, Tk-Instruct, and GPT-J. 111
7.5 **Prototypes instructions perform better**. 113
7.6 **LLMs predict many instances identically using instructions with *flipped* logic**, indicating they are overly biased. 114
7.7 **Few-shot examples provide larger benefits to LLMs *with* instructions than *without* instructions.** 115
7.8 **LLMs + Instructions are highly biased against specific examples.** Each dot is a single data point. Over 30 seeds, LLMs + Instructions with 4-shot examples consistently misclassify specific data points. 117

LIST OF TABLES

Page

3.1 Feature coefficients of LIME explanations for an instance from COMPAS, before and after an attack (ψ uses a single feature). 27

4.1 **Manipulated Models**: Test set accuracy and the size of the $\boldsymbol{\delta}$ vector for the four manipulated models (one for each counterfactual explanation algorithm), compared with the unmodified model trained on the same data. There is little change to accuracy using the manipulated models. Note, $\boldsymbol{\delta}$ is comparable across datasets due to unit variance scaling. 39

4.2 **Recourse Costs of Manipulated Models**: Counterfactual algorithms find similar cost recourses for both subgroups, however, give much lower cost recourse if $\boldsymbol{\delta}$ is added before the search. 40

5.1 **Evaluating Credible Intervals.** We report the % of time the 95% credible intervals with 100 perturbations include their true values (estimated on $10,000$ perturbations). Closer to 95.0 is better. Both BayesLIME and BayesSHAP are well calibrated. 60

6.1 A conversation about diabetes prediction, demonstrating the breadth of different conversation points the system can discuss. 80

6.2 Overview of the *operations* supported by TalkToModel, which are incorporated into the conversation to generate responses. 84

6.3 Exact Match Parsing Accuracy (%) for the 3 gold datasets, on the IID and Compositional splits, as well as Overall. The fine-tuned T5 models perform significantly better than few-shot GPT-J, and T5 Large performed the best. These results demonstrate that TalkToModel can understand user intentions with a high degree of accuracy using the T5 models. 88

6.4 The prediction gap on important features (PGI) and prediction gap on unimportant features (PGU) results. We bold the statistically significant best result. Overall, explanation selection is the best explanation method in all settings, except for PGU and the german credit data where it is better than SHAP but not significantly better than LIME. 93

6.5 User study results: % of respondents that agree (> Neutral Likert score) TalkToModel is better than the dashboard in the 4 comparison questions. A significant portion of respondents agreed TalkToModel is better than the dashboard in all the categories except Grad. students and "Likeliness To Use". Still, a majority agreed TalkToModel was superior in this case. 95

6.6 User study results: Completion rate and accuracy across interfaces and participant groups. We compute the completion rate as the questions users provided and answer for and did not mark "could not determine." We measure accuracy on completed questions. Participants answered questions at a higher rate more accurately using TalkToModel than the dashboard. 97

7.1 **Example Adult instance** (abbreviated), with title E_t, instructions I_t, classes C_t, and features F_t. 105

LIST OF ALGORITHMS

Page
1 Focused sampling for local explanations 59

ACKNOWLEDGMENTS

Foremost, I would like to say my deepest thank you to my advisor Sameer Singh. He has taught me so much about machine learning and research, but also how to be a better thinker, writer, and collaborator. These are invaluable gifts that will be useful wherever I go in the future. Moreover, his prescience in identifying promising research directions has helped me find impactful areas of research. Additionally, I have also always felt he takes my well-being to heart throughout my time working under his supervision and, in a field full of tight deadlines and reviewer #2's, this has always given me the confidence to keep moving forward. Some of my fondest memories during my Ph.D. are working towards deadlines, devising exciting new experiments, and finding clearer ways to present results with him—times like this were always quite thrilling!

I also want to sincerely thank my co-advisor Himabindu (Hima) Lakkaraju. Hima was a new professor the year I started my Ph.D. and Sameer, fortuitously, suggested we work on a project together. I'm so glad he did! Working with her has been a blessing during my Ph.D., and I owe her immensely for her countless hours spent revising and helping provide clarity to my research works. Hima's attention to detail, research vision, and ability to reason with reviewers is a great asset, and I am so grateful to have had the opportunity to collaborate with her on many projects during my Ph.D.

I also want to thank my undergraduate advisor, Sorelle Friedler, for introducing me to machine learning during my time at Haverford College. It has been a pleasure seeing her over years at conferences since my time and Haverford, and her ability to think about the greater context for research works has influenced me in many ways, and for this, I am extremely grateful. I always love to hear what she is thinking about because its usually something new and refreshing.

My time as a Ph.D. student has given me so much, and the colleagues and friends I made along the way are something I cherish. In no particular order (I randomized this), I want to especially thank: Caterina Belem, Charlie Marx, Richard Phillips, Gavin Kerrigan, Sakshi Agarwal, Yasaman Razeghi, Sam Showalter, Tamanna Hossain-Kay, Anthony Chen, Zhengli Zhao, Robbie Logan, Kolby Nottingham, Pouya Pezeshkpour, Shivanshu Gupta, Yanai Elazar, Federica Ricci, Markelle Kelly, Preethi Seshadri, Dheeru Dua, and Harry Bendekgey. From hanging out at UCI or at conferences to playing board games late at night, you have offered advice, provided your expertise, and made the process of getting a Ph.D. that much better.

I also want to thank my co-authors, without whom the work in this dissertation would not be possible: Sophie Hilgard, Emily Jia, Gavin Kerrigan, Jens Tuyls, Nathalie Rauschmayr, Krishnaram Kenthapadi, Muhammad Bilal Zafar, Yinlam Chow, Bo Dai, Nevan Wichers, Yuxin Chen, Chenhao Tan, and Satyapriya Krishna. I want to give special thanks to Sophie and Satya for working with me closely on many of my research works.

To my family, thank you for your immense support through my doctorate. To my brother, Trevor, thank you for always being up to chat about something interesting, whether its biology or rockets; I hope you, quite literally, make it to the moon! To my parents, Jessica and Phil, your endless guidance and the countless ways you've supported me have made my doctorate journey have made my life so much better. With your many FaceTimes or things you do to help me like taking care of Diego or fixing our heater, you've made this possible. What can I say to 26 years of unconditional love and support? I am immensely and forever grateful.

To Disi, I'm so glad I sent you that e-mail back during the first year of my Ph.D. to learn more about your research. Knowing you has been the happiest part of my life. From our home with Diego to the Mendocino coast line or whever else in the world we find ourselves, your appreciation of beautiful things in the world brings me so much joy. I look forward to your wit and candor everyday, and I love you deeply.

VITA

Dylan Slack

EDUCATION

Doctor of Philosophy in Computer Science **2023**
University of California, Irvine *Irvine, California*

Bachelor of Science in Computer Science **2019**
Haverford College *Haverford, Pennsylvania*

RESEARCH EXPERIENCE

Graduate Research Assistant **2019–2023**
University of California, Irvine *Irvine, California*

Research Scientist, Intern **2021**
Google *Remote*

Applied Scientist, Intern **2020**
Amazon Web Services (AWS) *Remote*

Research Assistant **2017-2019**
Haverford College *Haverford, Pennsylvania*

Research Assistant **2019**
Swarthmore College *Swarthmore, Pennsylvania*

GRANTS

Hasso Plattner Institute **2020-2022**
Fellowship

TEACHING EXPERIENCE

Teaching Assistant **2023**
CS 178

COMPSCI 282BR, Guest Lecture 2021
Havard University

Reader **2019**
CS 178

Academic Service

Reviewer
TMLR: 2023
NeurIPS: 2019, 2020, 2021 (*Top Reviewer*), 2022 (*Top Reviewer*)
ICLR: 2021 (*Outstanding Reviewer*)
FAcct: 2021
AAAI: 2020, 2021
ICML: 2020
KDD: 2019
Organizer
KDD Deep Learning Day: 2021

REFEREED JOURNAL PUBLICATIONS

Active Meta-Learning for Predicting and Selecting Perovskite Crystallization Experiments **2022**
Journal of Chemical Physics

Context, Language Modeling, and Multimodal Data in Finance **2021**
Journal of Financial Data Science

REFEREED CONFERENCE PUBLICATIONS

Reliable Post hoc Explanations: Modeling Uncertainty in Explainability **2021**
NeurIPS

Counterfactual Explanations Can Be Manipulated **2021**
NeurIPS

On the Lack of Robustness of Neural Text Classifier Interpretations **2021**
ACL

Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods **2020**
AIES

Fairness Warnings and Fair-MAML: Learning Fairly with Minimal Data **2020**
FAccT

REFEREED WORKSHOP PUBLICATIONS

TalkToModel: Explaining Machine Learning Models with Interactive Natural Language Conversations **2022**
TSRML @ NeurIPS, (*Honorable Mention Outstanding Paper*)

Rethinking Explainability as a Dialogue: A Practitioner's Perspective **2022**
HCAI @ NeurIPS

SAFER: Data-Efficient and Safe Reinforcement Learning via Skill Acquisition **2022**
DARL @ ICML

Defuse: Training More Robust Models through Creation and Correction of Novel Model Errors **2021**
XAI 4 Debugging @ NeurIPS

Differentially Private Language Models Benefit from Public Pre-training **2020**
PrivateNLP @ EMNLP

Assessing the Local Interpretability of Machine Learning Models **2019**
HCML @ NeurIPS

INVITED TALKS

TalkToModel: Explaining Machine Learning Models with Interactive Natural Language Conversations **2023**
HPI Launch Event

Exposing Shortcomings and Improving the Reliability of Machine Learning Explanations **2022**
Venues: Stanford MedAI Seminar Series, Imperial College London, Facebook AI, UC Irvine CML Seminar

Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods **2021**
Aggregate Intellect

Fairness Warnings and Fair-MAML: Learning Fairly with Minimal Data **2020**
FAccT Conference

ABSTRACT OF THE DISSERTATION

Robust Interactions with Machine Learning Models

By

Dylan Slack

Doctor of Philosophy in Computer Science

University of California, Irvine, 2023

Associate Professor Sameer Singh, Chair

Due to its strong predictive power, machine learning (ML) has increasingly shown considerable potential to disrupt a wide range of critical domains, such as medicine, healthcare, and finance. Along with this success, ML models have become more complex and parameter intensive. For instance, large language models (LLMs) pre-trained on massive amounts of internet text have become a default choice for many prediction problems. As a result, models are increasingly difficult to understand, establish trust in, and have become more data-intensive. To address the opaqueness of ML models, researchers have proposed *explanation* methods that help users understand why their models make predictions. Still, explanation methods often do not faithfully explain model predictions, and domain experts struggle to use them. As a result, it is important to understand how ML explanations fail, improve their robustness, and enhance their usability. Moreover, due to the increased data-intensiveness of many ML problems and the desire for widespread integration, there is a need for methods that achieve strong predictive performance more easily and cost-effectively.

In this dissertation, we address these problems in two main research thrusts: 1) We evaluate the shortcomings of explanation methods by developing *adversarial attacks* on such techniques, which provide insights into how these methods fall short. We propose novel explanation methods that are more robust to common issues these explanations suffer. 2) We develop language-based methods of interacting with explanations, enabling anyone to understand machine learning models. We extend these findings to a more general predictive setting where we improve model performance using natural language instructions to solve critical prediction tasks with only minimal training data.

First, we examine the limitations of explanation methods through the lens of adversarial attacks. We introduce adversarial attacks on two commonly used types of explanations: local post hoc explanations, and counterfactual explanations. Our methods reveal that it is possible to design ML models for whom explanations behave unfaithfully, demonstrating that they are not robust. We additionally analyze other limiting factors of explanations, such as their instability and inconsistency, and demonstrate how improved uncertainty quantification can alleviate these issues. To this end, we introduce two new explanation methods, including uncertainty estimates for explanations, *BayesLIME* and *BayesSHAP*, that overcome many of these robustness issues.

Second, we analyze the usability of current explanation methods and find that many subject matter experts, like healthcare workers or policy researchers, struggle to use them. To overcome these issues, we introduce TalkToModel: an interactive, natural language dialogue system for explaining ML models. Our real-world evaluations suggest TalkToModel dramatically helps improve the usability of ML explanations. Based on the finding that natural language is a highly useful interface between models and humans, we evaluate how well current LLMs utilize natural language instructions for solving tabular prediction tasks from instructions and introduce a benchmark of prediction tasks, *TABLET*, to this end. Taken together, these works offer new techniques for making ML models more accessible to end users through natural language.

Chapter 1

Introduction

Machine learning (ML) models have increasingly gained in performance over the past decade across a broad range of critical application areas such as medicine, finance, and recommendation systems. As a result of this success, there is an ever-growing demand to integrate ML systems into new applications. However, as models have gained in performance, they have also become more complex and thus harder for stakeholders and model designers to understand. As a result, it has become harder to determine when to trust ML models and verify they are making decisions for the right reasons, which is critical for ensuring their safe application in the wild.

To help overcome these issues, researchers have developed *explanation* methods for explaining the predictions of ML models in understandable terms to humans. For instance, *post hoc* explanation such as LIME and SHAP explain the predictions of a trained ML model by describing the features responsible for any given prediction [Ribeiro et al., 2016a, Lundberg and Lee, 2017c]. Similarly, *counterfactual* explanations describe minimal changes to instances needed to alter model predictions, providing means for recourse for adversely affected individuals [Wachter et al., 2018]. Because of their capacity to provide justifications for predictions, explanation methods have seen widespread adoptions to help people understand and establish trust in ML systems.

Because ML explanations play a critical role in establishing trust in ML systems, they also represent a significant source of vulnerability. Indeed, if explanations fail to adequately account for why a model makes a prediction or are fooled into concealing the true workings of a model by adversaries, model stakeholders could deploy faulty ML models. In high stakes settings, such as medicine and finance, deploying faulty ML models could lead to highly negative outcomes. Therefore, it is critical to understand where existing explanations fall short and develop more robust techniques.

In addition, implementing and understanding explanation techniques currently requires significant ML expertise, making it hard for domain experts to take advantage of these methods. Thus, beyond developing more robust explanation methods, there is also a need to develop techniques which make it easier for anyone to use explanation methods in practice. One of the more promising routes for supporting model explainability is the use of *natural language conversations*. Such conversations could allow anyone to understand ML models by simply engaging in an interactive dialogue, where they can ask open-ended questions about the ML system. Because natural language is a useful interface between models and people, it has considerable potential to improve other aspects of the model development process, such as training ML models.

As such, this disseration will address the following research questions:

1. In which ways are current state-of-the-art explanations not robust and how can we develop more robust explainability techniques?

2. How can we develop conversational systems that simplify the process of developing and explaining ML models for end users?

1.1 Contributions & Key Findings

In this section, we provide an overview of the main content of this dissertation and highlight the contributions within its two research thrusts.

1.1.1 Robust Explainability

In this subsection, we discuss the contributions of the first thrust: evaluating and improving the robustness of explanation methods. This thrust covers chapters 3, 4, and 5.

Chapters 3 & 4: Attacks on Post hoc and Counterfactual Explanations Explanations have seen widespread usage across many critical areas, such as medical and financial domains. Therefore, it is highly important to verify that they are trustworthy. In these chapters, we propose adversarial attacks on two broadly used types of explanations: local post hoc and counterfactual explanations. Local post hoc explanations describe which features are most influential to a prediction, thereby, expressing to users the features the model relied on most to arrive at a decision [Slack et al., 2021a, Lundberg and Lee, 2017a, Ribeiro et al., 2016a]. Counterfactual explanations provide minimal sets of modifications to data points that change the model's prediction, providing a means of recourse for individuals adversely affected by a model's decision [Wachter et al., 2018, Van Looveren and Klaise, 2019, Mothilal et al., 2020]. These works demonstrate that an adversary can design an ML model in such as way that both types of explanations *do not* reveal the underlying behavior of the model and instead present behaviors of the adversary's design. Practically, for the attack on feature importance explanations, we demonstrate that an adversary can design a highly unfair ML model based on sensitive attributes in the data (e.g., `race` or `gender`), but local explanations do not reveal this fact. Similarly, we show an adversary can design ML models that show equal cost recourse across demographic groups yet conceal unfairness in the model for one of the groups. For instance, when the adversary applies a small known perturbation to one of the groups, the recourse cost is much lower, indicating the model is actually unfair. Overall, we demonstrate two broadly used types of explanations are vulnerable to manipulation, indicating that further work is needed to improve their safety.

Chapter 5: Robust Post hoc Explanations While we have shown that local explanations are not robust to direct manipulation by adversaries, they also have several other sources of instability without direct interference. Local explanations are unstable—small perturbations to data points lead to significant changes in the explanation [Alvarez-Melis and Jaakkola, 2018] They also are highly inconsistent, and reruns lead to different explanations [Slack et al., 2021a]. Finally, it is challenging to determine key hyperparameters for explanations, such as how much to sample and where to sample the model when reasonable explanations. To help overcome these issues, this chapter proposes a novel Bayesian framework for generating explanations and their associated uncertainty. We instantiate this framework to obtain Bayesian versions of LIME and KernelSHAP, which output credible intervals for the feature importances, capturing the associated uncertainty. The resulting explanations enable us to make concrete inferences about their quality (e.g., there is a 95% chance that the feature importance lies within the given range) and are also highly consistent and stable. We carry out a detailed theoretical analysis that leverages the aforementioned uncertainty to estimate how many perturbations to sample and how to sample for faster convergence. This work makes the first attempt at addressing several critical issues with popular explanation methods in one shot, thereby generating consistent, stable, and reliable explanations with guarantees in a computationally efficient manner. Our experimental evaluation with multiple real-world datasets and user studies demonstrates the efficacy of the proposed framework.

1.1.2 Conversational Systems for ML Development

In this subsection, we discuss the second thrust of this dissertation: developing conversational systems for explainability and model development. This thrust covers chapters 6 and 7.

Chapter 6: Interactive Explanations While explanations are helpful for machine learning practitioners to understand ML models, it is unclear whether explanations meet the usability needs of critical subject matter experts, such as doctors or policy researchers. To begin Chapter 6, we address this gap and conduct a study where we interview doctors, healthcare professionals, and policymakers about their needs and desires for explanations. Our study indicates that decision-makers would strongly prefer interactive explanations. In particular, they would prefer these interactions to be natural language dialogues. Domain experts wish to treat machine learning models as "another colleague," i.e., one who can be held accountable by asking why they made a particular decision through expressive and accessible natural language interactions.

Chapter 6: Explainability Dialogues with TalkToModel Based on the findings of our previous study, we continue Chapter 6 by introducing a conversational system for explaining machine learning models called TalkToModel. Users can discuss with TalkToModel why predictions occur, how the predictions would change if the data changes, and how to flip predictions, among many other conversation topics. To support such rich conversations with TalkToModel, we introduce language understanding and model explainability techniques. First, we propose a dialogue engine that parses user text inputs (user utterances) into an SQL-like programming language using a large language model (LLM). The LLM performs the parsing by treating the task of translating user utterances into the programming language as a seq2seq learning problem, where the user utterances are the source and parses in the programming language are the targets. In our evaluations, we find TalkToModel understands users with a high degree of accuracy, and users have a much easier time explaining models using the system.

Chapter 7: Learning from Instructions with TABLET So far, we have shown enabling model explainability using natural language conversations can lead to much improved experiences for end users. Can we also use natural language to improve performance within a predictive setting? In this chapter, we study the use of *natural language instructions* for improving predictive performance on tabular datasets. While collecting tabular data for prediction tasks in medical or financial domains can be costly, we instead envision that domain experts could interactively provide natural language instructions to LLMs, which in turn could serve as tabular prediction models. To evaluate how well instructions leverage the knowledge in LLMs for improving predictive performance, we introduce TABLET: a benchmark of tabular prediction tasks annotated with natural language instructions. The tasks include many diverse domains, such as finance, healthcare, and medicine, and we collect several instruction annotations for each task that vary in their source, granularity, and phrasing. In our evaluation, we find LLMs can use instructions to perform well on tabular prediction tasks in the zero and few-shot shot settings, while models without instructions struggle. Still, LLMs underperform SOTA supervised learning models fit on the complete training set for the tasks and LLMs are often not faithful to instructions, demonstrating there are opportunities for researchers to develop models that improve zero and few-shot performance on TABLET.

1.2 Declaration of Previous Work and Collaborations

This dissertation is based on several published works, written under the supervision of Sameer Singh and Himabindu Lakkaraju. Particularly:

- The techniques for attacking local model agnostic explanations such as LIME and SHAP (Chapter 3) were published in the conference on Artificial Intelligence, Ethics and Society (AIES 2020) in collaboration with Sophie Hilgard and Emily Jia [Slack et al., 2020].

- The method for attacking counterfactual explanations (Chapter 4) was published in the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) in collaboration with Sophie Hilgard [Slack et al., 2021b].

- The bayesian framework for computing local model agnostic explanations and the corresponding BayesLIME and BayesSHAP techniques (Chapter 5) was published in the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) in collaboration with Sophie Hilgard [Slack et al., 2021a].

- The interviews motivating the need for model explainability with natural language conversations (Chapter 6) was presented at the workshop on human-centered AI at NeurIPS (HCAI @ NeurIPS 2022) in collaboration with Yuxin Chen and Chenhao Tan [Lakkaraju et al., 2022]. TalkToModel (Chapter 6) was presented at the workshop on Trustworthy and Socially Responsible Machine Learning at NeurIPS (TSRML @ NeurIPS 2022) in collaboration with Satyapriya Krishna where it won an honorable mention outstanding paper award, and is currently under submission at Nature Machine Intelligence.

- The TABLET benchmark for evaluating LLM ability to solve tabular prediction using natural language instructions (Chapter 7 is currently under submission at the Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)).

Part I

Robust Machine Learning Explanations

Chapter 2

Explainability Background

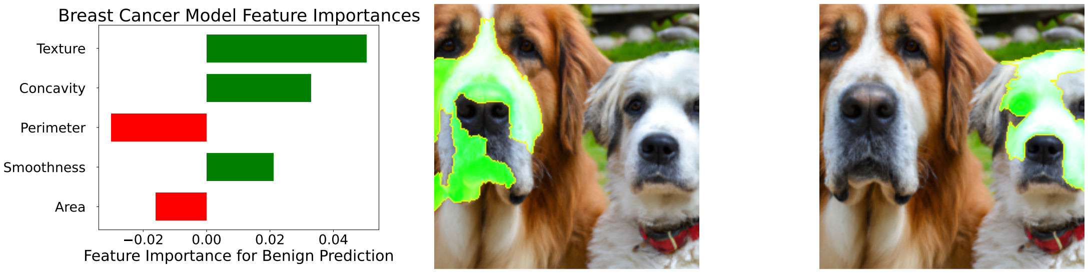
In this chapter, we motive and provide background for machine learning (ML) explanations. Owing to the success of machine learning (ML) models, there has been an increasing interest in leveraging these models to aid decision makers (e.g., doctors, judges) in critical domains such as healthcare and criminal justice. The successful adoption of these models in domain-specific applications relies heavily on how well decision makers are able to understand and trust their functionality [Doshi-Velez and Kim, 2017, Lipton, 2018]. Only if decision makers have a clear understanding of the model behavior, can they diagnose errors and potential biases in these models, and decide when and how much to rely on them. However, the proprietary nature and increasing complexity of machine learning models makes it challenging for domain experts to understand these complex *black boxes*, thus, motivating the need for tools that can explain them in a faithful and interpretable manner. As a result, there has been a recent surge in post hoc techniques for explaining black box models in a human interpretable manner, which are often used to help domain experts detect discriminatory biases in black box models [Tan et al., 2018, Kim et al., 2018].

2.1 Notation

Let $\mathcal{D}$ denote the input dataset of N data points i.e., $\mathcal{D} = (\mathcal{X}, \boldsymbol{y}) = \{(x_1, y_1), (x_2, y_2) \cdots (x_N, y_N)\}$ where x_i is a vector that captures the feature values of data point i, and y_i is the corresponding class label. Let there be M features in the dataset $\mathcal{D}$ and let $\mathcal{C}$ denote the set of class labels in $\mathcal{D}$ i.e., $y_i \in \mathcal{C}$. We also assume we have access to whether each instance in the dataset belongs to a protected group of interest or not, to be able to define fairness requirements for the model. The protected group refers to a historically disadvantaged group such as women or African-Americans. We use $\mathcal{D}_{\text{pr}}$ to indicate the protected subset of the dataset $\mathcal{D}$, and use $\mathcal{D}_{\text{np}}$ for the "not-protected" group. Further, we denote the protected group with a positive (i.e. *more desired*) outcome as $\mathcal{D}_{\text{pr}}^{\text{pos}}$ and with negative (i.e. *less desired*) outcome as $\mathcal{D}_{\text{pr}}^{\text{neg}}$ (and similarly for the non-protected group). Further, we have a model f that predicts the probability of a class y_i using a datapoint $f : x \rightarrow [0, 1]$. Finally, we assume the model is paramaterized by $\boldsymbol{\theta}$ but omit the dependence and write f for convenience. Last, we assume the positive class is the desired outcome (e.g., receiving a loan) henceforth.

2.2 Local Model Agnostic Explanations

One of the most prominent explanation techniques is *local, model-agnostic* methods that focus on explaining individual predictions of a given black box classifier, including **LIME** [Ribeiro et al., 2016b] and **SHAP** [Lundberg and Lee, 2017a]. Due to their generality, these methods have been used to explain a number of classifiers, such as neural networks and complex ensemble models, and in various domains ranging from law, medicine, finance, and science [Elshawi et al., 2019, Ibrahim et al., 2019, Whitmore et al., 2016]. Specifically, LIME and SHAP estimate feature attributions on individual instances, which capture the *contribution* of each feature on the black box prediction. Below, we provide some details of these approaches,



(a) Explaining `Benign` (b) Explaining `St. Bernard` (c) Explaining `Pyrenees`

Figure 2.1: Example LIME model agnostic explanations on a tabular and image classification problem, demonstrating local explanations for different data modalities and classes. Green indicates *positive* contribution for the model predicting the class, while red indicates *negative*. The leftmost graph (a) includes an explanation for a `benign` prediction for a random forest training to predict the presence of breast cancer. The center (b) and right (c) images are explanations for the `St. Bernard` and `Pyrenees` classes for the Inception V3 neural network.

while also highlighting how they relate to each other.

The goal of model agnostic local explanations is to explain individual predictions of f in an interpretable and faithful manner. Note that neither LIME nor SHAP assume any knowledge about the internal workings of f. Let g denote an explanation model that we intend to learn to explain f, g is a linear model $g(x) = \phi^T x$.

Let $\pi_x(x')$ denote the proximity measure between inputs x and x', to define the vicinity (neighborhood) around x. Let $\mathcal{Z}$ be a set of N randomly sampled instances (perturbations) around x. We denote the vector of these distances over the N perturbations in $\mathcal{Z}$ as $\Pi_x(\mathcal{Z}) \in \mathbb{R}^N$. Let $Y \in [0,1]$ be the vector of the black box predictions $f(z)$ corresponding to each of the N instances in $\mathcal{Z}$. With all this notation in place, the objective function for both LIME and SHAP is crafted to generate an explanation that approximates the behavior of the black box accurately within the vicinity of x with an interpretable linear model.

$$\arg\min_{g \in \mathcal{G}} L(f, g, \pi_x) \tag{2.1}$$

where the loss function L is defined as:

$$L(f, g, \pi_x) = \sum_{z \in \mathcal{Z}} [f(z) - \phi^T z]^2 \pi_x(z) \tag{2.2}$$

where X' is the set of inputs constituting the neighborhood of x. The above objective function has the following closed form solution:

$$\hat{\phi} = (\mathcal{Z}^T \mathrm{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + \mathbb{I})^{-1}(\mathcal{Z}^T \mathrm{diag}(\Pi_x(\mathcal{Z}))Y) \tag{2.3}$$

The primary difference between LIME and SHAP lies in how π_x is chosen. In LIME, this functions is defined heuristically: $\pi_x(x')$ is defined using cosine or $l2$ distance. On the other hand, (Kernel) SHAP grounds these definitions in game theoretic principles to guarantee that the explanations satisfy desired properties, like that the feature importances for the explanations sum up to the predicted value. More details about the intuition behind the definitions of these functions and their computation can be found in Ribeiro et al. [2016b] and Lundberg and Lee [2017a]. We provide practical examples in Figure 2.1.

2.3 Counterfactual Explanations

While local explanations help understand *why* a model makes a particular decision, they do not explicitly reveal *what needs to change* to get a different outcome for a prediction. As a result, there are a growing number of methods that explain the decisions of these models to affected individuals and provide means for *recourse* [Ustun et al., 2019]. For example, recourse offers a person denied a loan by a credit risk model a reason for *why* the model made the prediction and *what can be done* to change the decision. Beyond providing guidance to stakeholders in model decisions, algorithmic recourse is also used to detect discrimination in machine learning models [Gupta et al., 2019, Karimi et al., 2020, Sharma et al., 2020]. For instance, we expect there to be minimal *disparity* in the *cost* of achieving recourse between both men and women who are denied loans. One commonly used method to generate recourse is that of *counterfactual explanations* [Bhatt et al., 2020]. Counterfactual explanations offer recourse by attempting to find the minimal change an individual must make to receive a positive outcome [Wachter et al., 2018, Karimi et al., 2020, Poyiadzi et al., 2020, Van Looveren and Klaise, 2019].

Counterfactual explanations return a data point that is *close* to $\boldsymbol{x}$ but is predicted to be positive by the model f. We denote the counterfactual returned by a particular algorithm $\mathcal{A}$ for instance $\boldsymbol{x}$ as $\mathcal{A}(\boldsymbol{x})$ where the model predicts the positive class for the counterfactual, i.e., $f(\mathcal{A}(\boldsymbol{x})) > 0.5$. We take the difference between the original data point $\boldsymbol{x}$ and counterfactual $\mathcal{A}(\boldsymbol{x})$ as the set of changes an individual would have to make to receive the desired outcome. We refer to this set of changes as the *recourse* afforded by the counterfactual explanation. We define the *cost* of recourse as the *effort* required to accomplish this set of changes [Venkatasubramanian and Alfano, 2020]. In this work, we define the cost of recourse as the distance between $\boldsymbol{x}$ and $\mathcal{A}(\boldsymbol{x})$. Because computing the real-world cost of recourse is challenging [Barocas et al., 2020], we use an ad-hoc distance function, as is general practice.

Counterfactual Objectives In general, counterfactual explanation techniques optimize objectives of the form,

$$G(\boldsymbol{x}, \boldsymbol{x}_{\text{cf}}) = \lambda \cdot (f(\boldsymbol{x}_{\text{cf}}) - 1)^2 + d(\boldsymbol{x}, \boldsymbol{x}_{\text{cf}}) \tag{2.4}$$

in order to return the counterfactual $\mathcal{A}(\boldsymbol{x})$, where $\boldsymbol{x}_{\text{cf}}$ denotes *candidate* counterfactual at a particular point during optimization. The first term $\lambda \cdot (f(\boldsymbol{x}_{\text{cf}}) - 1)$ encourages the counterfactual to have the desired outcome probability by the model. The distance function $d(\boldsymbol{x}, \boldsymbol{x}_{\text{cf}})$ enforces that the counterfactual is close to the original instance and easier to "achieve" (lower cost recourse). λ balances the two terms. Further, when used for algorithmic recourse,

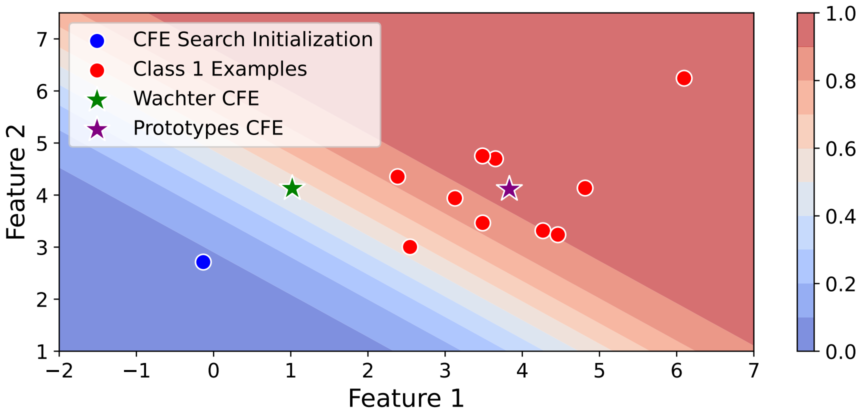


Figure 2.2: We compare two choices of distance functions on the search for counterfactual explanations (CFEs) on a toy problem. The start of the search for the CFE is the point in blue within the negative classification region. The Wachter et. al's distance function (green) emphasizes closeness to the search initialization and the CFE being classified positively. The distance function from Van Looveren and Klaise [2019] (purple) additionally emphasizes *distributionality* of the CFE, meaning it should be more realistic for achieving. While both these algorithms find valid CFEs, they are quite different because the objectives emphasize different desirable properties of CFEs.

counterfactual explainers often only focus on the few features that the user can influence in the search and the distance function; we omit this in the notation for clarity.

Distance Functions The distance function $d(\boldsymbol{x}, \boldsymbol{x}_{\text{cf}})$ captures the *effort* needed to go from $\boldsymbol{x}$ to $\boldsymbol{x}_{\text{cf}}$ by an individual. As one such notion of distance, Wachter et al. [2018] use the Manhattan (ℓ_1) distance weighted by the inverse median absolute deviation (MAD).

$$d(\boldsymbol{x}, \boldsymbol{x}_{\text{cf}}) = \sum_{q \in [d]} \frac{|\boldsymbol{x}^q - \boldsymbol{x}^q_{\text{cf}}|}{\text{MAD}_q} \qquad \text{MAD}_q = \text{median}_{i \in [N]} \left(|x^q_i - \text{median}_{j \in [N]}(x^q_j)| \right) \tag{2.5}$$

This distance function generates sparse solutions and closely represents the absolute change someone would need to make to each feature, while correcting for different ranges across the features. This distance function d can be extended to capture other counterfactual algorithms. For instance, we can include elastic net regularization instead of ℓ_1 for more efficient feature selection in high dimensions [Dhurandhar et al., 2018], add a term to capture the closeness of the counterfactual $\boldsymbol{x}_{\text{cf}}$ to the data manifold to encourage the counterfactuals to be in distribution, making them more realistic [Van Looveren and Klaise, 2019], or include diversity criterion on the counterfactuals [Mothilal et al., 2020]. We compare some of these distance functions in Figure 2.2.

Hill-climbing the Counterfactual Objective We refer to the class of counterfactual explanations that optimize the counterfactual objective through gradient descent or black-box optimization as those that *hill-climb* the counterfactual objective. For example, Wachter et al.'s algorithm [Wachter et al., 2018] or DiCE [Mothilal et al., 2020] fit this characterization because they optimize the objective in Equation 2.4 through gradient descent. Methods like MACE [Karimi et al., 2020] and FACE [Poyiadzi et al., 2020] do not fit this criteria because they do not use such techniques.

Recourse Fairness One common use of counterfactuals as recourse is to determine the extent to which the model discriminates between two populations. For example, counterfactual explanations may return recourses that are easier to achieve for members of the not-protected group [Ustun et al., 2019, Sharma et al., 2020] indicating unfairness in the counterfactuals [Karimi et al., 2020, Gupta et al., 2019]. Formally, we define the recourse fairness as the difference in the average distance of the recourse cost between the protected and not-protected groups, and we say a counterfactual algorithm $\mathcal{A}$ is *recourse fair* if this disparity is less than some threshold τ.

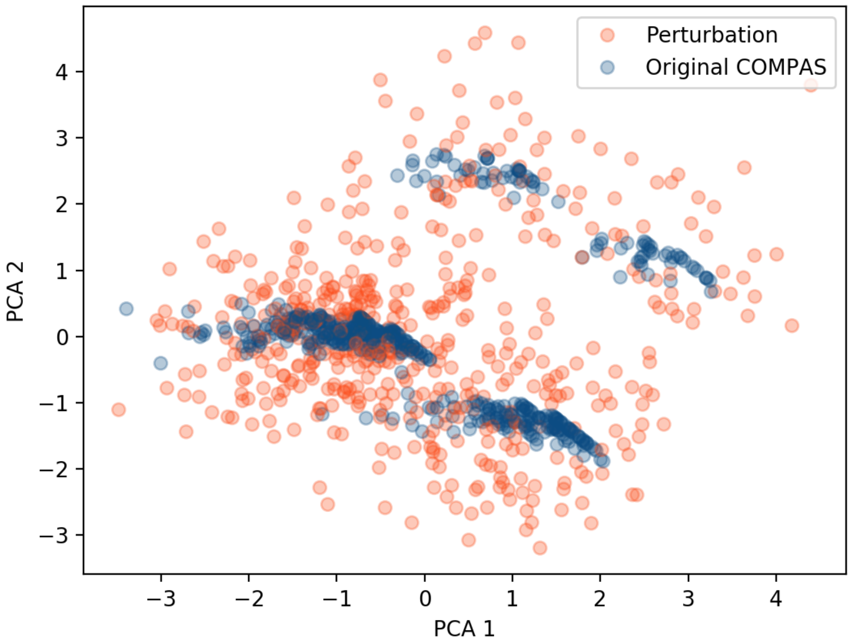
Definition 2.1. *A model f is recourse fair for algorithm $\mathcal{A}$, distance function d, dataset $\mathcal{D}$, and scalar threshold τ if [Gupta et al., 2019],*

$$\left|\mathbb{E}_{x\sim\mathcal{D}_{pr}^{neg}}\left[d\left(\boldsymbol{x}, \mathcal{A}(\boldsymbol{x})\right)\right] - \mathbb{E}_{x\sim\mathcal{D}_{np}^{neg}}\left[d\left(\boldsymbol{x}, \mathcal{A}(\boldsymbol{x})\right)\right]\right| \leq \tau$$

Chapter 3

Adversarial Attacks On Local Post Hoc Explanations

In this chapter, we demonstrate significant vulnerabilities in post hoc explanation techniques that can be exploited by an adversary to generate classifiers whose post hoc explanations can be arbitrarily controlled. More specifically, we develop a novel framework that can effectively mask the discriminatory biases of any black box classifier. Our approach exploits the fact that post hoc explanation techniques such as LIME [Ribeiro et al., 2016b] and SHAP [Lundberg and Lee, 2017a] are perturbation-based, to create a *scaffolding* around any given biased black box classifier in such a way that its predictions on input data distribution remain biased, but its behavior on the perturbed data points is controlled to make the post hoc explanations look completely innocuous. For instance, using our framework, we generate highly discriminatory scaffolded classifiers (such as the ones that *only* use race to make their decisions) whose post hoc explanations (generated by LIME and SHAP) make them look completely innocuous, effectively hiding their discriminatory biases.



l

Figure 3.1: PCA applied to the COMPAS dataset (blue) as well as its LIME style perturbations (red). Even in this low-dimensional space, we can see that data points generated via perturbations are distributed very differently from instances in the COMPAS data. In this paper, we exploit this difference to craft adversarial classifiers.

3.1 Proposed Framework

In this section, we discuss our framework in detail. First, we discuss some preliminary details about our set up. Then, we discuss the intuition behind our approach. Lastly, we present the technical details of our approach along with a discussion of some of our design choices and implementation details.

Preliminaries

Setting: Assume that there is an adversary with an incentive to deploy a biased classifier f for making a critical decision (e.g., parole, bail, credit) in the real world. The adversary must provide black box access to customers and regulators [Regulation, 2016], who may use post hoc explanation techniques to better understand f and determine if f is ready to be used in the real world. If customers and regulators detect that f is biased, they are not likely to approve it for deployment. The goal of the adversary is to fool post hoc explanation techniques and hide the underlying biases of f.

Input: The adversary provides the following to our framework: 1) the biased classifier f which they intend to deploy in the real world and, 2) an input dataset $\mathcal{X}$ that is sampled from the real world input data distribution $\mathcal{X}_{dist}$ on which f will be applied. Note that neither our framework nor the adversary has access to $\mathcal{X}_{dist}$.

Output: The output of our framework will be a scaffolded classifier e (referred to as *adversarial classifier* henceforth) that behaves exactly like f when making predictions on instances sampled from $\mathcal{X}_{dist}$, but will not reveal the underlying biases of f when probed with leading post hoc explanation techniques such as LIME and SHAP.

Intuition As discussed in the previous section, LIME and SHAP (and several other post hoc explanation techniques) explain individual predictions of a given black box model by constructing local interpretable approximations (e.g., linear models). Each such local approximation is designed to capture the behavior of the black box within the neighborhood of a given data point. These neighborhoods constitute synthetic data points generated by perturbing features of individual instances in the input data. However, instances generated using such perturbations could potentially be off-manifold or out-of-distribution (OOD) [Mittelstadt et al., 2019].

To better understand the nature of the synthetic data points generated via perturbations, we carried out the following experiment. First, we perturb input instances using the approach employed by LIME (See previous section). We then run principal component analysis (PCA) on the combined dataset containing original instances as well as the perturbed instances, and reduce the dimensionality to 2. As we can see from Figure 3.1, the synthetic data points generated from input perturbations are distributed significantly differently from the instances in the input data. This result indicates that detecting whether a data point is a result of a perturbation or not is not a challenging task, and thus approaches that rely heavily on these perturbations, such as LIME, can be *gamed*.

This intuition underlies our proposed approach. By being able to differentiate between data points coming from the input distribution and instances generated via perturbation, an adversary can create an adversarial classifier (*scaffolding*) that behaves like the original classifier (perhaps be extremely discriminatory) on the input data points, but behaves arbitrarily differently (looks unbiased and *fair*) on the perturbed instances, thus effectively fooling LIME or SHAP into generating innocuous explanations. Next, we formalize this intuition and explain our framework for building adversarial classifiers that can fool explanation techniques.

Building Adversarial Classifiers Let f be the biased classifier described earlier. The adversary would like to design a framework such that if and when end users generate explanations of this black box, post hoc techniques can be fooled into thinking that the model is innocuous. Recall that the real world data on which this classifier is likely to be applied follows a distribution $\mathcal{X}_{dist}$, and $\mathcal{X}$ is a set of N data points sampled from this distribution that the adversary has access to. To fool the post hoc techniques, the adversary could design an adversarial classifier that exhibits biased behavior on instances sampled from $\mathcal{X}_{dist}$, and remain unbiased on instances that do not come from $\mathcal{X}_{dist}$. Since the feature importances output by LIME and SHAP rely heavily on perturbed instances (which may typically be OOD samples, e.g. Figure 3.1), the resulting explanations will make the classifier designed by the adversary look innocuous.

Assuming ψ is a *unbiased* classifier (e.g., makes predictions based on innocuous features that are uncorrelated with sensitive attributes), the adversarial classifier e takes the following form:

$$e(x) = \begin{cases} f(x), & \text{if } x \in \mathcal{X}_{dist} \\ \psi(x), & \text{otherwise} \end{cases} \tag{3.1}$$

In order to create such a classifier, we need to be able to decide whether a given data point x comes from $\mathcal{X}_{dist}$ or not.

Detecting OOD Samples To build a classifier `is_OOD` that detects if a given data point is an *out-of-distribution* (OOD) sample (`is_OOD`$(x) =$ `True`) or not, we construct a new dataset from the instances in $\mathcal{X}$. Specifically, we *perturb* (more details in Experimental Evaluation) each instance x in $\mathcal{X}$ to generate a new instance x_p. Let us denote the set of all the instances generated via perturbation as $\mathcal{X}_p$. The instances in $\mathcal{X}$ are then assigned the class label `False` indicating that they are not OOD samples, while the instances in $\mathcal{X}_p$ are assigned the class label `True` (indicating that they are OOD samples) unless they are already in $\mathcal{X}$. We then train an off-the-shelf classifier on the combined dataset $\mathcal{X} \cup \mathcal{X}_p$ and their corresponding class labels (assigned as discussed above).

3.2 Experimental Results

In this section, we discuss the detailed experimental evaluation of our framework. First, we analyze the effectiveness of the adversarial classifiers generated by our framework. More specifically, we test how well these classifiers can mask their biases by fooling multiple post hoc explanation techniques. Next, we evaluate the robustness of our adversarial classifiers by measuring how their effectiveness varies with changes to different parameters (e.g., weighting kernel, background distribution). Lastly, we present examples of post hoc explanations (both LIME and SHAP) of individual instances in the data to demonstrate how the biases of the classifier f are successfully hidden.

Datasets We experimented with multiple datasets pertaining to diverse yet critical real world applications such as recidivism risk prediction, violent crime prediction, and credit scoring. Our first dataset is the **COMPAS** dataset which was collected by ProPublica Angwin et al. [2016]. This dataset captures detailed information about the criminal history, jail and prison time, demographic attributes, and COMPAS risk scores for 6172 defendants from Broward Couty, Florida. The sensitive attribute in this dataset is race – 51.4% of the defendants are African-American. Each defendant in the data is labeled either as high-risk or low-risk for recidivism. Our second dataset is **Communities and Crime** (CC) that captures various socio-economic and law enforcement aspects, as well as crime across various communities in the US Redmond and Baveja [2002]. This dataset contains information 1994 communities (each community is a data point) in total. The sensitive attribute in this dataset is the percentage of Caucasian population. This percentage varies widely across communities and ranges from 2.68% to 99.63%. Each community is assigned a label depending on if the violent crime proportion in that community is above the median (violent crime rampant) or not. The third dataset we use is the **German credit** dataset from the UCI repository Blake et al. [1999]. This dataset captures financial and demographic information (including account information, credit history, employment, gender) of about 1000 loan applicants. The sensitive attribute in this dataset is gender – 69% of the individuals are male. Each individual in the dataset is labeled either as a good customer or a bad one depending on their credit risk.

Experimental Setup Here, we discuss the implementation details of our framework. We randomly split all the datasets into training (90%) and test (10%) sets. We build our framework using the training set and evaluate it on the test set.

Biased Classifier f: We construct f to be a perfectly discriminatory classifier that uses a sensitive feature i.e., f makes predictions purely based on race in the case of COMPAS and Communities and Crime datasets (if $race = African\ American$, then predict 1, otherwise set predict 0), and based on gender for the German Credit dataset.

Perturbations & OOD classifier: For LIME, we generate perturbations for every data point in the data set $\mathcal{X}$ by adding random noise sampled from $\mathcal{N}(0, 1)$ to each feature value. For SHAP, we randomly choose a subset of features for each data point and mark their values as missing by replacing them with their corresponding values from background distribution[1].We choose the background distribution for SHAP to be cluster centers from kmeans with 10 clusters. We use these newly generated perturbations as well as the input data points to train the OOD classifier to detect OOD samples (See Proposed Framework section). We use the standard sklearn implementation of a random forest with 100 tree estimators as our OOD classifier.

Unbiased Classifier ψ: We build the unbiased classifier ψ by constructing synthetic *uncorrelated features* that have zero correlation with sensitive attributes (e.g., race or gender). We experiment with one or two uncorrelated features. When we only have one uncorrelated feature in a particular experiment, ψ solely uses that to make predictions (if uncorrelated feature = 1, then predict 1, else predict 0). On the other hand, when we have two uncorrelated features in an experiment, we base the predictions on the xor of those two features. Note that ψ does not have to be restricted to always use synthetic uncorrelated features. It can also use any other existing feature in the data to make predictions. We experiment with synthetic uncorrelated features on COMPAS and CC dataset, and with *Loan Rate % Income* feature on the German credit dataset.

Generating Explanations: We use default LIME tabular implementation without discretization, and the default Kernel SHAP implementation with kmeans with 10 clusters as the background distribution.

---

[1] See Lundberg and Lee [2017a] for more details.

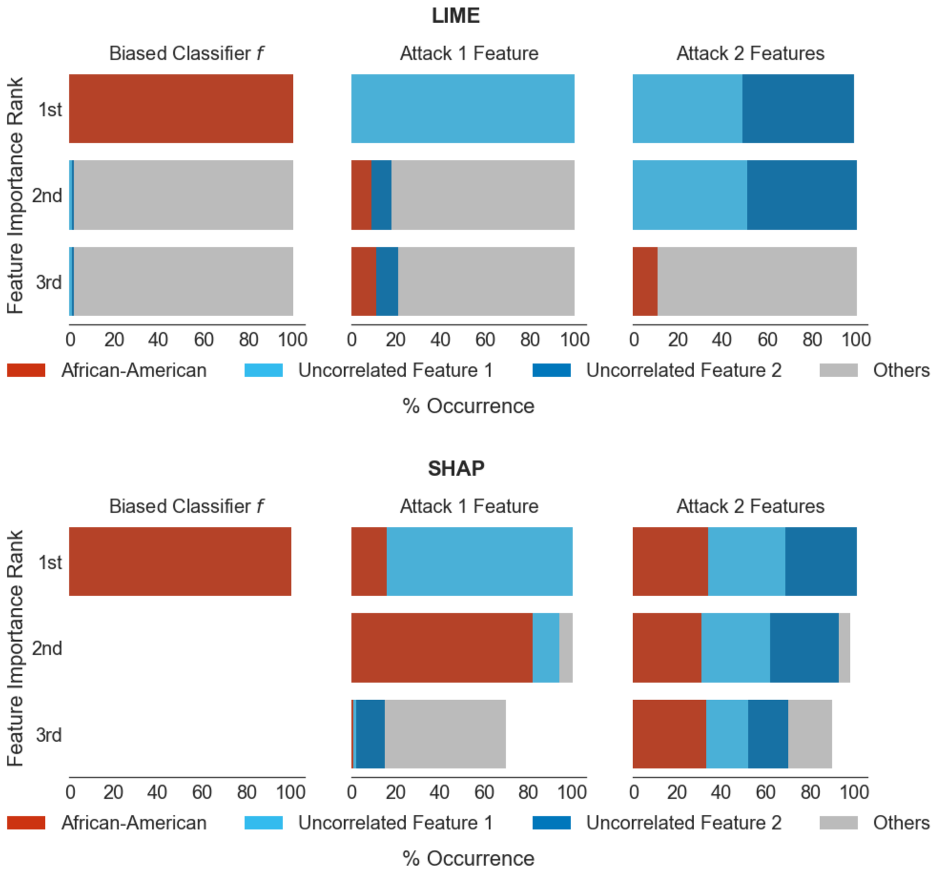


Figure 3.2: **COMPAS:** % of data points for which each feature (color coded) shows up in top 3 (according to LIME and SHAP's ranking of feature importances) for the biased classifier f (left), our adversarial classifier where ψ uses only one uncorrelated feature to make predictions (middle), and our adversarial classifier where ψ uses two uncorrelated features to make predictions (right).

3.2.1 Effectiveness of Adversarial Classifiers

To evaluate how successful our *attacks* are on LIME and SHAP, we compute the percentage of data points for which race, uncorrelated features (in case of COMPAS and CC) or *Loan Rate % Income* (in case of German credit data) show up in top 3 when features are ranked based on feature attributions output by LIME and SHAP. In figures 3.2, 3.3, and 3.4, we show the results of these experiments. In case of **LIME**, when a single feature is used for the attack i.e., ψ uses a single feature for making predictions, the adversarial classifier e successfully shifts the feature importance in LIME from the true informative feature (the sensitive feature

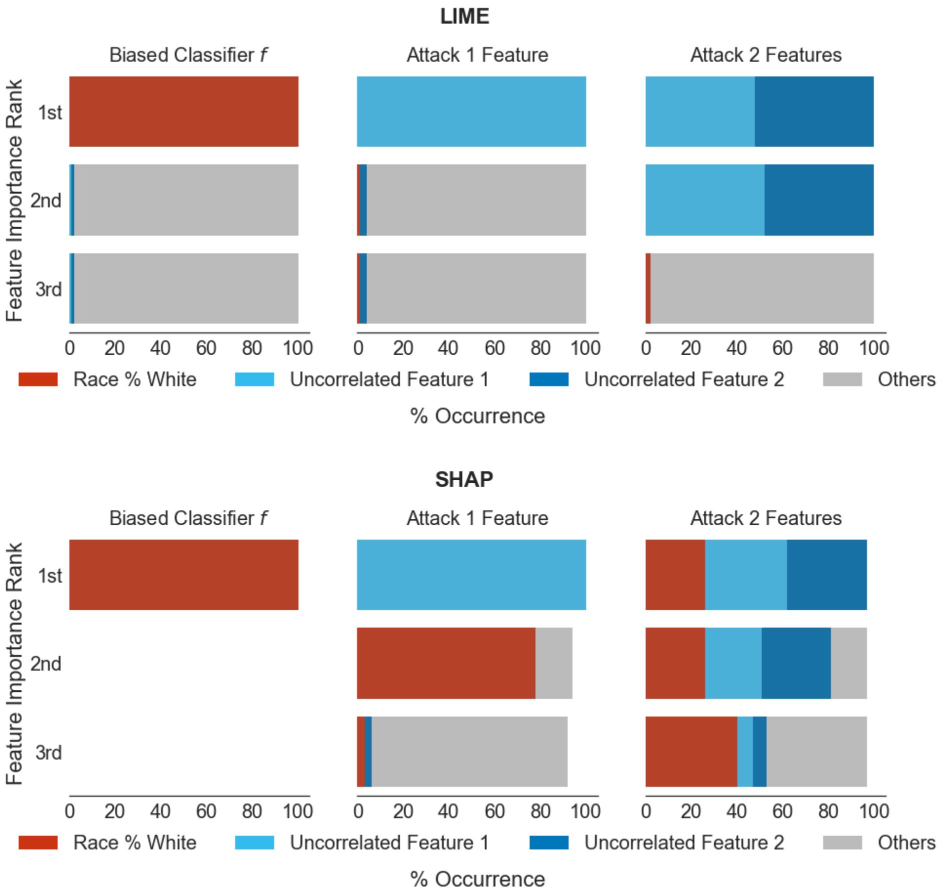


Figure 3.3: **Communities and Crime:** Similar to Fig 3.2; *Race % White* is the sensitive feature here.

used by f) to the uncorrelated feature in 100% of held-out points for COMPAS (Figure 3.2 - top and middle) and CC (Figure 3.3 - top and middle), and to the *Loan % Income* feature in 91% for German (Figure 3.4 - middle). Furthermore, the sensitive feature appears at 2nd or 3rd rank position roughly only as often as other features. When both uncorrelated features are used to define ψ (for COMPAS and CC), the 1st and 2nd rank positions for nearly all the data points are occupied by these features, with the sensitive feature entering into 3rd place at about the same rate as other features (Figures 3.2 & 3.3 - top right). These results demonstrate that the LIME explanation technique has been effectively fooled by the adversarial classifier e for all three datasets.

For **SHAP**, when a single uncorrelated feature is used for the attack, the adversarial classifier

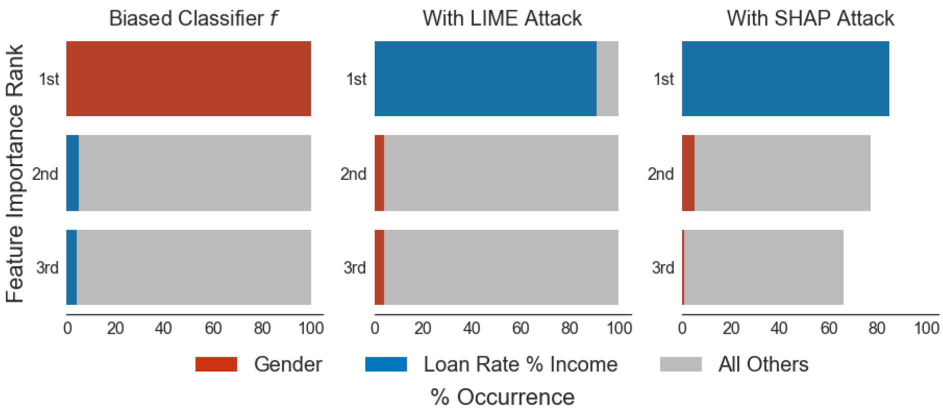
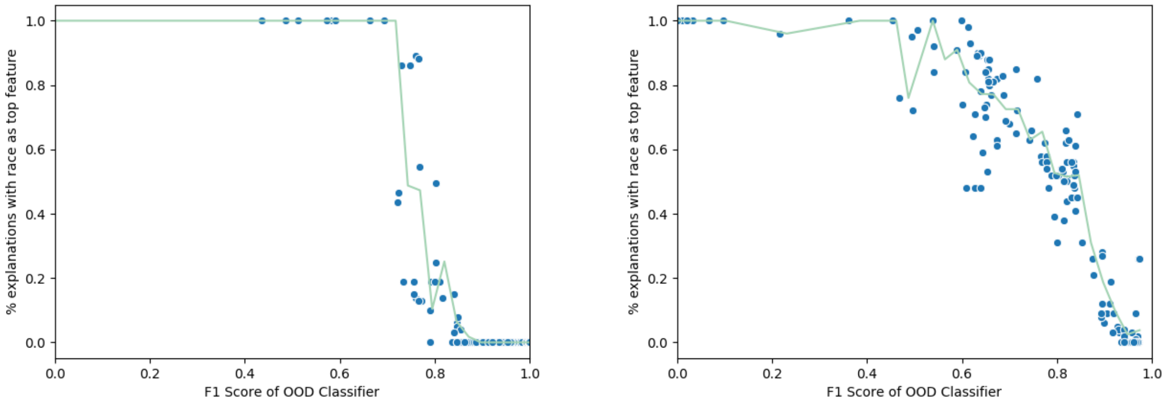


Figure 3.4: **German credit:** Similar to Fig 3.2 and 3.3, but unbiased classifier ψ uses an existing feature (*Loan Rate % Income*) to make predictions, and *Gender* is the sensitive feature. Feature importances for the biased classifier f shown in the figure (left) are generated using LIME; SHAP also produces similar feature importance values.

e successfully shifts the feature importance from the sensitive feature in 100% of data points in CC (Figure 3.3 - bottom and middle), 84% of data points in COMPAS (Figure 3.2 - bottom and middle), and 85% of data points in German. When two uncorrelated features are used in the attack (for COMPAS and CC), the adversarial classifier is less successful in removing the bias feature from 1st place in the ranking, succeeding in only 67% and 71% instances for COMPAS and CC respectively (Figures 3.2 & 3.3 - bottom right). This is due to SHAP's local accuracy property that ensures that feature attributions must add up to the difference between a given prediction and the average prediction for the background distribution. This property will tend to distribute feature attributions across several features when it is not possible to identify a single most informative feature.

To ensure that the adversarial classifiers e are faithfully approximating the original classifiers f on instances from the original dataset, we evaluate the *fidelity* of e w.r.t. f by computing the percentage of held-out instances on which their predictions agree. In case of LIME, our adversarial classifiers perfectly reproduce the classifier f on all datasets 100% of the time. On the other hand, in case of SHAP, our adversarial classifiers are slightly less effective: 75% for CC, 80% for German, and 91% for COMPAS. These results indicate that our approach



(a) LIME COMPAS Sensitivity Analysis

(b) SHAP COMPAS Sensitivity Analysis

Figure 3.5: Effectiveness of our attack: % of data points for which race is the most important feature vs. the F1 score of the OOD classifier (perturbation detection) for both LIME and SHAP on the COMPAS dataset over a variety of models (fit with line plot over mean of binned points).

has constructed adversarial classifiers e that successfully fool LIME and SHAP, making them look like the unbiased ψ classifiers, while behaving like the original biased classifiers f on input data.

3.2.2 Effect of Perturbation Detection Accuracy

Our approach relies on the OOD classifier to detect whether a given instance is a result of perturbation or not. Here, we investigate how accurate does this classifier have to be for our attacks to be effective. We train a number of different OOD classifiers with varying accuracies (F1 scores) on the COMPAS dataset. For each OOD classifier, we evaluate the effectiveness of the resulting attacks (measured as percentage of data points for which race is the most important feature in the post hoc explanations). The results are presented in Figure 3.5. When the OOD classifiers are unable to adequately differentiate between perturbed instances and points from the input dataset, both LIME and SHAP explanations detect race as the most important feature i.e., the adversarial attacks are not successful. However, once the OOD classifier is more accurate, the adversarial attacks are successful, and both LIME and

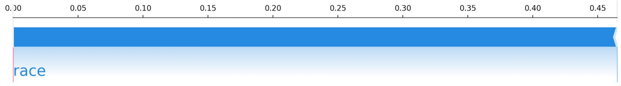
Feature	**Before**	**After attack**
Race	-0.42	0.0
Uncorrelated feature 1	0.0	0.42
Average of other features	0.0	0.0

Table 3.1: Feature coefficients of LIME explanations for an instance from COMPAS, before and after an attack (ψ uses a single feature).

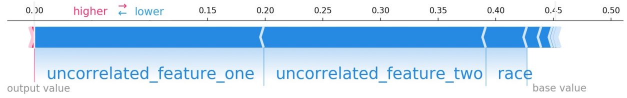
SHAP do not identify race as the most important feature. Between the two, LIME requires the OOD classifier to be more accurate for the attack to be effective, but is very vulnerable to these attacks once the F1 score is above 0.8. SHAP, on the other hand, starts responding to the attacks with a less accurate OOD classifier (F1 score $\sim$ 0.45), however, the effect is more gradual; it needs a very accurate OOD classifier to completely hide the bias. These results indicate that our attacks are effective as long as it is possible to differentiate between perturbed instances and input data points with a reasonable accuracy.

3.2.3 Robustness to Hyperparameters

We analyze the behavior of our framework when different parameters are varied *at test time* when generating LIME and SHAP explanations for COMPAS; we summarize the key results here. First, we observe that classifiers that are adversarial to LIME are ineffective against SHAP explanations, while adversarial classifiers designed for SHAP have varying effectiveness against LIME. On varying the LIME kernel width between $0.15\sqrt{M}$ to $0.95\sqrt{M}$ (default kernel width is $0.75\sqrt{M}$, where M is the number of features), the attacks remain equally effective. Additionally, changing the LIME kernel to use $l1$ norm for distance instead of the default $l2$ norm also does not impact the effectiveness. For SHAP, we train our adversarial classifier assuming the default background distribution of K-means with 10 clusters; 5 and 15 clusters at test-time are also equally effective. Using a background distribution of all zeroes for SHAP makes our attacks even more pronounced.



(a) Example SHAP explanation for biased classifier f



(b) SHAP explanation of the same instance after the attack (e)

Figure 3.6: Example SHAP explanations for a data point with biased classifier f (top) and adversarial classifier e (bot.).

3.2.4 Example Explanations

Here, we present a few example explanations from the COMPAS dataset that demonstrate the effectiveness of the technique. In case of LIME explanations in Table 3.1, we see that *Race* as a feature is completely hidden from the explanation after the attack, even though it was the sole important feature for the original classifier f. Similarly, for SHAP explanations in Figure 3.6, the sole important feature (*race*) is considerably hidden in the explanation after the attack, although not completely nullified as in the LIME explanation.

3.3 Summary of Contributions

In this chapter, we proposed a novel framework that can effectively hide discriminatory biases of any black box classifier. Our approach exploits the fact that post hoc explanation techniques such as LIME and SHAP are perturbation-based to create a *scaffolding* around the discriminative classifier such that its predictions on input data distribution remain biased. However, its behavior on the perturbed data points makes the post hoc explanations look entirely innocuous. Extensive experimentation with real-world data from criminal justice and credit scoring domains demonstrates that our approach is effective at generating adversarial classifiers that can fool post hoc explanation techniques, finding that LIME is more vulnerable than SHAP. Our findings thus suggest that existing post hoc explanation techniques are insufficient for ascertaining classifiers' discriminatory behavior in sensitive applications. This chapter is based on the following publication:

- *Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods* (Slack et al. [2020], AIES 2020)

which has 487 citations at the time of writing.

The author of this dissertation was responsible for the results in this work. The author credits the shared first author of this work, Sophie Hilgard, with much of the writing and framing of the work.

Chapter 4

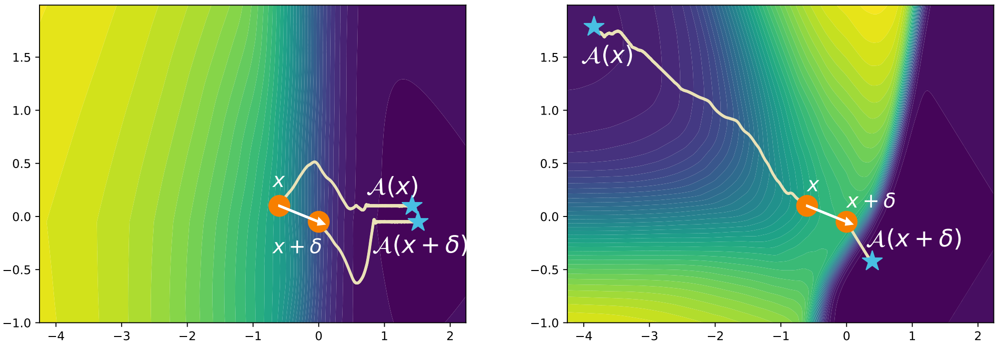
Adversarial Attacks on Counterfactual Explanations

The previous chapter demonstrated that adversaries can fool post hoc explanations, demonstrating that these explanations are not robust. While post hoc explanations are broadly adopted, several other impactful types of explanations exist, such as *counterfactual explanations*. Because explanations play a significant role in establishing trust in ML models, it is additionally worthwhile investigating whether adversaries can similarly fool counterfactual explanations.

In this chapter, we introduce a novel framework that describes how counterfactual explanation techniques are not robust.[1] More specifically, we demonstrate how the family of counterfactual explanations that rely on hill-climbing (which includes commonly used methods like Wachter et al.'s algorithm [Wachter et al., 2018], DiCE [Mothilal et al., 2020], and counterfactuals guided by prototypes [Van Looveren and Klaise, 2019]) is highly sensitive to small changes in the input. To demonstrate how this shortcoming could lead to negative consequences,

---

[1] Note, that the usage of "counterfactual" does not have the same meaning as it does in the context of causal inference, and we adopt the term "counterfactual explanation" for consistency with prior literature.



(a) **Training with BCE Objective** (b) **Training Adversarial Model**

Figure 4.1: **Model trained with BCE objective and adversarial model on a toy data set** using Wachter et al.'s Algorithm [Wachter et al., 2018]. The surface shown is the loss in Wachter et al.'s Algorithm with respect to $\boldsymbol{x}$, the line is the path of the counterfactual search, and we show results for a single point, $\boldsymbol{x}$. For the model without the manipulation (subfigure 4.1a), the counterfactuals for $\boldsymbol{x}$ and $\boldsymbol{x}+\boldsymbol{\delta}$ converge to the same minima and are similiar cost recourse. For the adversarial model (subfigure 4.1b), the recourse found for $\boldsymbol{x}$ has *higher cost* than $\boldsymbol{x}+\boldsymbol{\delta}$ because the local minimum initialized at $\boldsymbol{x}$ is *farther* than the minimum starting at $\boldsymbol{x}+\boldsymbol{\delta}$, demonstrating the problematic behavior of counterfactual explanations.

we show how these counterfactual explanations are vulnerable to manipulation. Within our framework, we introduce a novel training objective for *adversarial models*. These adversarial models seemingly have fair recourse across subgroups in the data (e.g., men and women) but have much lower cost recourse for the data under a slight perturbation, allowing a bad-actor to provide low-cost recourse for specific subgroups simply by adding the perturbation. To illustrate the adversarial models and show how this family of counterfactual explanations is not robust, we provide two models trained on the same toy data set in Figure 4.1. In the model trained with the standard BCE objective (left side of Fig 4.1), the counterfactuals found by Wachter et al.'s algorithm [Wachter et al., 2018] for instance $\boldsymbol{x}$ and perturbed instance $\boldsymbol{x}+\boldsymbol{\delta}$ converge to same minima (denoted $\mathcal{A}(\boldsymbol{x})$ and $\mathcal{A}(\boldsymbol{x}+\boldsymbol{\delta})$). However, for the adversarial model (right side of Fig 4.1), the counterfactual found for the perturbed instance $\boldsymbol{x}+\boldsymbol{\delta}$ is *closer* to the original instance $\boldsymbol{x}$. This result indicates that the counterfactual found for the perturbed instance $\boldsymbol{x}+\boldsymbol{\delta}$ is *easier to achieve* than the counterfactual for $\boldsymbol{x}$ found by Wachter et al.'s algorithm! Intuitively, counterfactual explanations that hill-climb the gradient are susceptible to this issue because optimizing for the counterfactual at $\boldsymbol{x}$ versus

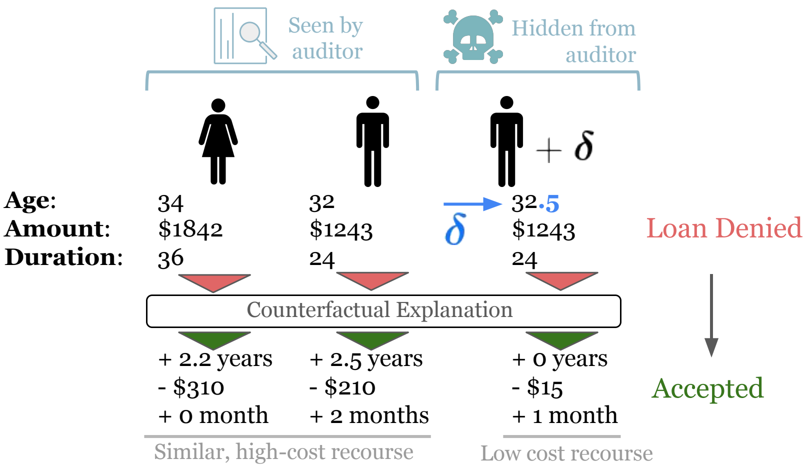


Figure 4.2: **Manipulated Model for Loan Risk.** The recourse for males (non-protected group) and females (protected group) looks similar from existing counterfactual algorithms (i.e. model seems fair). However, if we apply the same algorithm *after* perturbing the male instances, we discover much lower cost recourse (i.e. the model discriminates between sexes).

$\boldsymbol{x} + \boldsymbol{\delta}$ can converge to different local minima.

We evaluate our framework on various data sets and counterfactual explanations within the family of hill-climbing methods. For Wachter et al.'s algorithm [Wachter et al., 2018], a sparse variant of Wachter et al.'s, DiCE [Mothilal et al., 2020], and counterfactuals guided by prototypes [Van Looveren and Klaise, 2019], we train models on data sets related to loan prediction and violent crime prediction with fair recourse across subgroups that return 2-20× lower cost recourse for specific subgroups with the perturbation $\boldsymbol{\delta}$, without any accuracy loss. Though these results indicate counterfactual explanations are highly vulnerable to manipulation, we consider making counterfactual explanations that hill-climb the gradient more robust. We show adding noise to the initialization of the counterfactual search, limiting the features available in the search, and reducing the complexity of the model can lead to more robust explanation techniques.

4.1 Adversarial Models for Manipulating Counterfactual Explanations

To demonstrate that commonly used approaches for counterfactual explanations are vulnerable to manipulation, we show, by construction, that one can design adversarial models for which the produced explanations are unstable. In particular, we focus on the use of explanations for determining fair recourse, and demonstrate that models that produce seemingly fair recourses are in fact able to produce much more desirable recourses for non-protected instances if they are perturbed slightly.

Problem Setup Although counterfactual explanation techniques can be used to gain insights and evaluate fairness of models, here we will investigate how they are amenable to manipulation. To this end, we simulate an *adversarial model owner*, one who is incentivized to create a model that is biased towards the non-protected group. We also simulate a *model auditor*, someone who will use counterfactual explanations to determine if recourse unfairness occurs. Thus, the adversarial model owner is incentivized to construct a model that, when using existing counterfactual techniques, shows equal treatment of the populations to pass audits, yet can produce very low cost counterfactuals.

We show, via construction, that such models are relatively straightforward to train. In our construction, we jointly learn a *perturbation vector* $\boldsymbol{\delta}$ (a small vector of the same dimension as $\boldsymbol{x}$) and the model f, such that the recourses computed by existing techniques look fair, but recourses computed by adding perturbation $\boldsymbol{\delta}$ to the input data produces low cost recourses. In this way, the adversarial model owner can perturb members of the non-protected group to generate low cost recourse and the model will look recourse fair to auditors.

Motivating Example For a concrete example of a real model that meets this criteria, we refer to Figure 4.2. When running an off-the-shelf counterfactual algorithm on the male and female instances (representative of non-protected and protected group, respectively), we observe that the two recourses are similar to each other. However, when the adversary changes the age of the male applicant by 0.5 years (the perturbation $\boldsymbol{\delta}$), the recourse algorithm finds a much lower cost recourse.

Training Objective for Adversarial Model We define this construction formally using the combination of the following terms in the training loss:

- *Fairness:* We want the counterfactual algorithm $\mathcal{A}$ to be fair for model f according to Definition 2.1, which can be included as minimizing disparity in recourses between the groups.
- *Unfairness:* A perturbation vector $\boldsymbol{\delta} \in \mathbb{R}^d$ should lead to lower cost recourse when added to non-protected data, leading to unfairness, i.e., $\mathbb{E}_{x \sim \mathcal{D}_{\text{pr}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}(\boldsymbol{x})\right)\right] \gg \mathbb{E}_{x \sim \mathcal{D}_{\text{np}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}(\boldsymbol{x} + \boldsymbol{\delta})\right)\right]$.
- *Small perturbation:* Perturbation $\boldsymbol{\delta}$ should be *small*. i.e. we need to minimize $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{np}}^{\text{neg}}} d(\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{\delta})$.
- *Accuracy:* We should minimize the classification loss $\mathcal{L}$ (such as cross entropy) of the model f .
- *Counterfactual:* $(\boldsymbol{x} + \boldsymbol{\delta})$ should be a counterfactual, so that running $\mathcal{A}(\boldsymbol{x} + \boldsymbol{\delta})$ returns a counterfactual close to $(\boldsymbol{x} + \boldsymbol{\delta})$, i.e. minimize $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{np}}^{\text{neg}}} \left(f(\boldsymbol{x} + \boldsymbol{\delta}) - 1\right)^2$.

This combined training objective is defined over both the parameters of the model $\boldsymbol{\theta}$ and the perturbation vector $\boldsymbol{\delta}$. Apart from requiring dual optimization over these two variables, the objective is further complicated as it involves $\mathcal{A}$, a black-box counterfactual explanation approach. We address these challenges in the next section.

Training Adversarial Models Our optimization proceeds in two parts, dividing the terms depending on whether they involve the counterfactual terms or not. First, we optimize the perturbation $\boldsymbol{\delta}$ and model parameters $\boldsymbol{\theta}$ on the subset of the terms that do not depend on the counterfactual algorithm, i.e. optimizing accuracy, counterfactual, and perturbation size[2]:

$$\boldsymbol{\delta} := \arg\min_{\boldsymbol{\delta}} \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{np}}^{\text{neg}}} \left(f(\boldsymbol{x}+\boldsymbol{\delta}) - 1\right)^2 + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{np}}^{\text{neg}}} d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta}) \tag{4.1}$$

Second, we optimize parameters $\boldsymbol{\theta}$, fixing the perturbation $\boldsymbol{\delta}$. We still include the classification loss so that the model will be accurate, but also terms that depend on $\mathcal{A}$ (we use $\mathcal{A}_{\boldsymbol{\theta}}$ to denote $\mathcal{A}$ uses the model f parameterized by $\boldsymbol{\theta}$). In particular, we add the two competing recourse fairness related terms: reduced disparity between subgroups for the recourses on the original data and increasing disparity between subgroups by generating lower cost counterfactuals for the protected group when the perturbation $\boldsymbol{\delta}$ is added to the instances. This objective is,

$$\boldsymbol{\theta} := \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) + \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{np}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta})\right)\right] + \left(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{pr}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{np}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right]\right)^2$$

$$\text{s.t.} \quad \mathbb{E}_{x\sim\mathcal{D}_{\text{np}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta})\right)\right] < \mathbb{E}_{x\sim\mathcal{D}_{\text{pr}}^{\text{neg}}} \left[d\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right] \tag{4.2}$$

Optimizing this objective requires computing the derivative (Jacobian) of the counterfactual explanation $\mathcal{A}_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}$, $\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})$. Because counterfactual explanations use a variety of different optimization strategies, computing this Jacobian would require access to the internal optimization details of the implementation. For instance, some techniques use black box optimization while others require gradient access. These details may vary by implementation or even be unavailable. Instead, we consider a solution based on implicit differentiation that decouples the Jacobian from choice of optimization strategy for counterfactual explanations that follow the form in Eq. (2.4). We calculate the Jacobian as follows:

Lemma 4.1. *Assuming the counterfactual explanation $\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})$ follows the form of the objective*

---

[2] The objectives discussed in this section use the training set, whereas, evaluation is done on a held out test set everywhere else.

in Equation 2.4, $\frac{\partial}{\partial \boldsymbol{x}_{cf}} G(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})) = 0$, and m is the number of parameters in the model, we can write the derivative of counterfactual explanation $\mathcal{A}$ with respect to model parameters $\boldsymbol{\theta}$ as the Jacobian,

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\left[\frac{\partial^2 G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)}{d\boldsymbol{x}_{cf}^2}\right]^{-1} \cdot \left[\frac{\partial}{\partial \boldsymbol{\theta}_1} \frac{\partial}{\partial \boldsymbol{x}_{cf}} G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \cdots \frac{\partial}{\partial \boldsymbol{\theta}_m} \frac{\partial}{\partial \boldsymbol{x}_{cf}} G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right]$$

Proof. We want to compute the derivative,

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{\theta}} \left[\arg\min_{\boldsymbol{x}_{\text{cf}}} G(\boldsymbol{x}, \boldsymbol{x}_{\text{cf}})\right] \tag{4.3}$$

This problem is identical to a well-studied class of bi-level optimization problems in deep learning. In these problems, we must compute the derivative of a function with respect to some parameter (here $\boldsymbol{\theta}$) that includes an inner argmin, which itself depends on the parameter. We follow [Gould et al., 2016] to complete the proof.

Note, we write $G(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}))$ to describe the objective G evaluated at the counterfactual found using the counterfactual explanation $\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})$. Also, we denote the zero vector as $\mathbf{0}$. For a single network parameter $\boldsymbol{\theta}_i$, $i \in \{1, ..., m\}$ we have the following equivalence because $\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})$ converges to a stationary point from the assumption,

$$\frac{\partial}{\partial \boldsymbol{x}_{\text{cf}}} G(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\text{cf}})) = \mathbf{0} \tag{4.4}$$

We differentiate with respect to $\boldsymbol{\theta}_i$ and apply the chain rule,

$$\frac{\partial}{\partial \boldsymbol{\theta}_i}\frac{\partial}{\partial \boldsymbol{x}_{\mathrm{cf}}}G(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})) + \frac{\partial^2}{\partial \boldsymbol{x}_{\mathrm{cf}}^2}G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})\right)\frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}}) = \mathbf{0} \tag{4.5}$$

$$\frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}}) = -\left[\frac{\partial^2}{\partial \boldsymbol{x}_{\mathrm{cf}}^2}G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})\right)\right]^{-1}\frac{\partial}{\partial \boldsymbol{\theta}_i}\frac{\partial}{\partial \boldsymbol{x}_{\mathrm{cf}}}G(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})) \tag{4.6}$$

Rewriting in terms of $\mathcal{A}$,

$$\frac{\partial}{\partial \boldsymbol{\theta}_i}\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\left[\frac{\partial^2}{\partial \boldsymbol{x}_{\mathrm{cf}}^2}G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})\right)\right]^{-1}\frac{\partial}{\partial \boldsymbol{\theta}_i}\frac{\partial}{\partial \boldsymbol{x}_{\mathrm{cf}}}G(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})) \tag{4.7}$$

Extending this result to multiple parameters, we write,

$$\frac{\partial}{\partial \boldsymbol{\theta}}\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\left[\frac{\partial^2 G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})\right)}{\partial \boldsymbol{x}_{\mathrm{cf}}^2}\right]^{-1}\left[\frac{\partial}{\partial \boldsymbol{\theta}_1}\frac{\partial}{\partial \boldsymbol{x}_{\mathrm{cf}}}G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})\right)\cdots\frac{\partial}{\partial \boldsymbol{\theta}_m}\frac{\partial}{\partial \boldsymbol{x}_{\mathrm{cf}}}G\left(\boldsymbol{x}, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})\right)\right] \tag{4.8}$$

□

This result depends on the assumption $\frac{\partial}{\partial \boldsymbol{x}_{\mathrm{cf}}}G(x, \mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x})) = 0$. This assumption states the counterfactual explanation $\mathcal{A}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{cf}})$ converges to a stationary point. In the case the counterfactual explanation terminates before converging to stationary point, this solution will be approximate.

Critically, this objective does not depend on the implementation details of counterfactual explanation $\mathcal{A}$, but only needs black box access to the counterfactual explanation. One potential issue is the matrix inversion of the Hessian. Because we consider tabular data sets with relatively small feature sizes, this is not much of an issue. For larger feature sets, taking the diagonal approximation of the Hessian has been shown to be a reasonable approximation

[Fernando and Gould, 2016, Bertsekas and Gallager, 1992].

To provide an intuition as to how this objective exploits counterfactual explanations to train manipulative models, we refer again to Figure 4.1. Because the counterfactual objective G relies on an arbitrary function f, this objective can be non-convex. As a result, we can design f such that G converges to higher cost local minimums for all datapoints $\boldsymbol{x} \in \mathcal{D}$ than those G converges to when we add $\boldsymbol{\delta}$.

4.2 Experiment Setup

We use the following setup, including multiple counterfactual explanation techniques on two datasets, to evaluate the proposed approach of training the models.

Counterfactual Explanations We consider four different counterfactual explanation algorithms as the choices for $\mathcal{A}$ that hill-climb the counterfactual objective. We use ***Wachter et al.***'s Algorithm [Wachter et al., 2018], Wachter et al.'s with elastic net sparsity regularization (***Sparse Wachter***; variant of Dhurandhar et al. [2018]), ***DiCE*** [Mothilal et al., 2020], and Counterfactual's Guided by ***Prototypes*** [Van Looveren and Klaise, 2019]. These counterfactual explanations are widely used to compute recourse and assess the fairness of models [Karimi et al., 2020, Verma et al., 2020, Stepin et al., 2021]. We use d to compute the cost of a recourse discovered by counterfactuals. We use the official DiCE implementation[3], and reimplement the others. DiCE is the only approach that computes multiple counterfactual explanations; we generate 4 counterfactuals and take the closest one to the original point (as per ℓ_1 distance) to get a single counterfactual.

Data sets We use two data sets: ***Communities and Crime*** and the ***German Credit*** datasets [Dua and Graff, 2017], as they are commonly used benchmarks in both the counter-

---

[3] `https://github.com/interpretml/DiCE`

Table 4.1: **Manipulated Models**: Test set accuracy and the size of the $\boldsymbol{\delta}$ vector for the four manipulated models (one for each counterfactual explanation algorithm), compared with the unmodified model trained on the same data. There is little change to accuracy using the manipulated models. Note, $\boldsymbol{\delta}$ is comparable across datasets due to unit variance scaling.

	Comm. & Crime		**German Credit**	
	Acc	$\|\|\boldsymbol{\delta}\|\|_1$	Acc	$\|\|\boldsymbol{\delta}\|\|_1$
Unmodified	81.2	-	71.1	-
Wachter et al.	80.9	0.80	72.0	0.09
Sparse Wachter	77.9	0.46	70.5	2.50
Prototypes	79.2	0.46	69.0	2.21
DiCE	81.1	1.73	71.2	0.09

factual explanation and fairness literature [Verma et al., 2020, Friedler et al., 2019]. Both these datasets are in the public domain. Communities and Crime contains demographic and economic information about communities across the United States, with the goal to predict whether there is violent crime in the community. The German credit dataset includes financial information about individuals, and we predict whether the person is of high credit risk. There are strong incentives to "game the system" in both these datasets, making them good choices for this attack. In communities and crime, communities assessed at higher risks for crime could be subject to reduced funding for desirable programs, incentivizing being predicted at low risk of violent crime [McGarry, 2012], while in German credit, it is more desirable to receive a loan. We preprocess the data as in Slack et al. [2020], and apply 0 mean, unit variance scaling to the features and perform an 80/20 split on the data to create training and testing sets. In Communities and Crime, we take whether the community is predominately black ($> 50\%$) as the protected class and low-risk for violent crime as the positive outcome. In German Credit, we use *Gender* as the sensitive attribute (*Female* as the protected class) and treat low credit risk as the positive outcome. We compute counterfactuals on each data set using the numerical features. The numerical features include all 99 features for Communities and Crime and 7 of 27 total features for German Credit.

Manipulated Models We use feed-forward neural networks as the adversarial model

Table 4.2: **Recourse Costs of Manipulated Models**: Counterfactual algorithms find similar cost recourses for both subgroups, however, give much lower cost recourse if $\boldsymbol{\delta}$ is added before the search.

	Communities and Crime				**German Credit**			
	Wach.	**S-Wach.**	**Proto.**	**DiCE**	**Wach.**	**S-Wach.**	**Proto.**	**DiCE**
Protected	35.68	54.16	22.35	49.62	5.65	8.35	10.51	6.31
Non-Protected	35.31	52.05	22.65	42.63	5.08	8.59	13.98	6.81
Disparity	*0.37*	*2.12*	*0.30*	*6.99*	*0.75*	*0.24*	*0.06*	*0.5*
Non-Protected+$\boldsymbol{\delta}$	1.76	22.59	8.50	9.57	3.16	4.12	4.69	3.38
Cost reduction	*20.1*×	*2.3*×	*2.6*×	*4.5*×	*1.8*×	*2.0*×	*2.2*×	*2.0*×

consisting of 4 layers of 200 nodes with the tanh activation function, the Adam optimizer, and using cross-entropy as the loss $\mathcal{L}$. It is common to use neural networks when requiring counterfactuals since they are differentiable, enabling counterfactual discovery via gradient descent [Mothilal et al., 2020]. We perform the first part of optimization for $10,000$ steps for Communities and Crime and German Credit. We train the second part of the optimization for 15 steps. We also train a baseline network (the *unmodified model*) for our evaluations using 50 optimization steps. In Table 4.1, we show the model accuracy for the two datasets (the manipulated models are similarly accurate as the unmodified one) and the magnitude of the discovered $\boldsymbol{\delta}$.

4.3 Experiments

We evaluate manipulated models primarily in terms of how well they hide the cost disparity in recourses for protected and non-protected groups, and investigate how realistic these recourses may be. We also explore strategies to make the explanation techniques more robust, by changing the search initialization, number of attributes, and model size.

4.3.1 Effectiveness of the Manipulation

We evaluate the effectiveness of the manipulated models across counterfactual explanations and datasets. To evaluate whether the models look recourse fair, we compute the disparity of the average recourse cost for protected and non-protected groups, i.e. Definition (2.1). We also measure the average costs (using d) for the non-protected group and the non-protected group perturbed by $\boldsymbol{\delta}$. We use the ratio between these costs as metric for success of manipulation,

$$\text{Cost reduction} := \frac{\mathbb{E}_{x\sim\mathcal{D}_{\text{np}}^{\text{neg}}}\left[d(\boldsymbol{x}, \mathcal{A}(\boldsymbol{x}))\right]}{\mathbb{E}_{x\sim\mathcal{D}_{\text{np}}^{\text{neg}}}\left[d(\boldsymbol{x}, \mathcal{A}(\boldsymbol{x}+\boldsymbol{\delta}))\right]}. \tag{4.9}$$

If the manipulation is successful, we expect the non-protected group to have much lower cost with the perturbation $\boldsymbol{\delta}$ than without, and thus the cost reduction to be high.

We provide the results for both datasets in Table 4.2. The disparity in counterfactual cost on the unperturbed data is very small in most cases, indicating the models would appear counterfactual fair to the auditors. At the same time, we observe that the cost reduction in the counterfactual distances for the non-protected groups after applying the perturbation $\boldsymbol{\delta}$ is quite high, indicating that lower cost recourses are easy to compute for non-protected groups. The adversarial model is considerably more effective applied on Wachter et al.'s algorithm in Communities and Crime. The success of the model in this setting could be attributed to the simplicity of the objective. The Wachter et al. objective only considers the squared loss (i.e., Eq (2.4)) and ℓ_1 distance, whereas counterfactuals guided by prototypes takes into account closeness to the data manifold. Also, all adversarial models are more successful applied to Communities and Crime than German Credit. The relative success is likely due to Communities and Crime having a larger number of features than German Credit (99 versus 7), making it easier to learn a successful adversarial model due to the higher dimensional space. Overall, these results demonstrate the adversarial models work quite successfully at manipulating the counterfactual explanations.

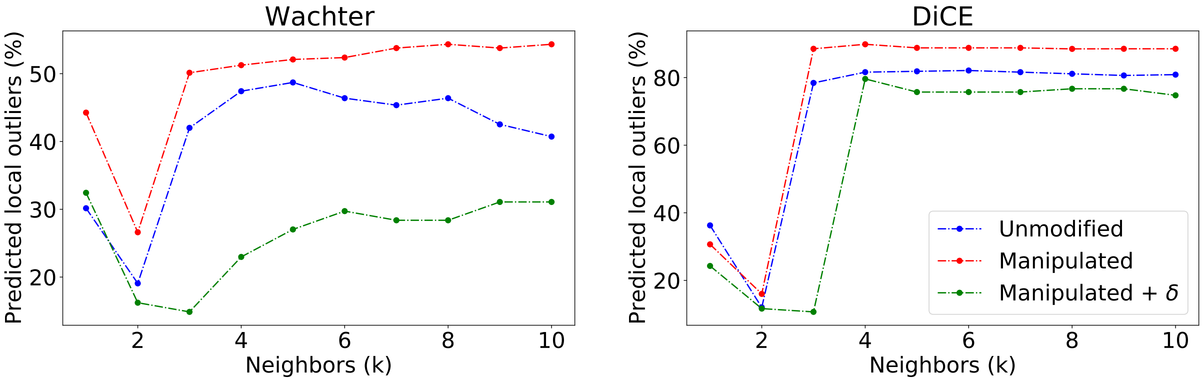


Figure 4.3: **Outlier Factor of Counterfactuals:** For the Wachter et al.'s and DiCE models for Communities and Crime, we show that the manipulated recourses are only slightly less realistic than counterfactuals of the unmodified model, whereas the counterfactuals found after adding $\boldsymbol{\delta}$ are more realistic than the original counterfactuals (lower is better).

4.3.2 Outlier Factor of Counterfactuals

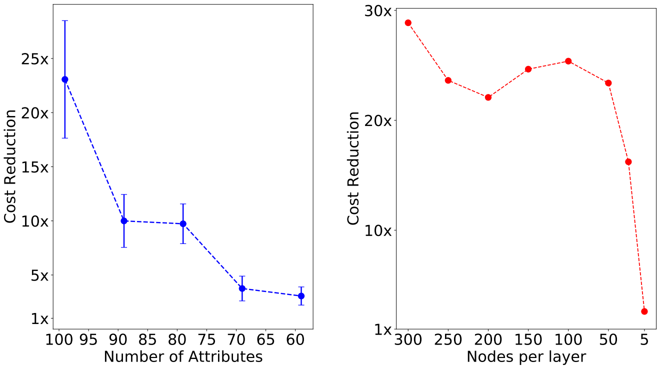
One potential concern is that the manipulated models returns counterfactuals that are out of distribution, resulting in unrealistic recourses. To evaluate whether this is the case, we follow Pawelczyk et al. [2020], and compute the local outlier factor of the counterfactuals with respect to the positively classified data [Breunig et al., 2000]. The score using a single neighbor ($k = 1$) is given as,

$$P(\mathcal{A}(\boldsymbol{x})) = \frac{d(\mathcal{A}(\boldsymbol{x}), a_0)}{\min_{\boldsymbol{x}\neq a_0 \in \mathcal{D}_{\text{pos}} \cap \{\forall x \in \mathcal{D}_{\text{pos}} | f(x)=1\}} d(a_0, \boldsymbol{x})}, \tag{4.10}$$

where a_0 is the *closest* true positive neighbor of $\mathcal{A}(\boldsymbol{x})$. This metric will be > 1 when the counterfactual is an outlier. We compute the percent of counterfactuals that are local outliers by this metric on Communities and Crime, in Figure 4.3. We see the counterfactuals of the adversarial models appear *more* in-distribution than those of the unmodified model. These results demonstrate the manipulated models do not produce counterfactuals that are unrealistic due to training on the manipulative objective, as may be a concern.

Model	**Wachter et al.**			**DiCE**
Initialization	Mean	Rnd.	$\boldsymbol{x}+\mathcal{N}$	Rnd.
Protected	42.4	16.2	11.7	48.3
Not-Prot.	42.3	15.7	10.3	42.3
Disparity	*0.01*	*0.49*	*1.45*	*5.95*
Not-Prot.+$\boldsymbol{\delta}$	2.50	3.79	8.59	12.3
Cost reduction	*16.9×*	*4.3×*	*1.2×*	*3.4×*
Accuracy	81.4	80.2	75.3	78.9
$\|\|\boldsymbol{\delta}\|\|_1$	0.65	0.65	0.36	1.24

(a) **Search Initialization:** Adding noise to the input is effective, at the cost to accuracy.



(b) **N. Features:** Fewer features make the manipulation less effective.

(c) **Model Size:** Smaller models are more effective at hiding their biases.

Figure 4.4: **Exploring Mitigation Strategies:** For the Wachter et al. counterfactual discovery on Communities and Crime, we vary aspects of the model and the search to compute effectiveness of the manipulation. Each provides a potentially viable defense, with different trade-offs.

4.3.3 Potential Mitigation Strategies

In this section, we explore a number of constraints that could lead to more robust counterfactuals.

Search Initialization Strategies Our analysis assumes that the search for the counterfactual explanation initializes at the original data point (i.e., $\boldsymbol{x}$ or $\boldsymbol{x}+\boldsymbol{\delta}$), as is common in counterfactual explanations. Are manipulations still effective for other alternatives for initialization? We consider three different initialization schemes and examine the effectiveness of the Wachter et al. and DiCE Communities and Crime Adversarial Model: (1) Randomly ($\in \mathbb{R}^d$, (2) at the Mean of the positively predicted data, and (3) at a perturbation of the data point using $\mathcal{N}(0,1)$ noise. To initialize Wachter et al. randomly, we follow Mothilal et al. [2020] and draw a random instance from a uniform distribution on the maximum and minimum of each feature (DiCE provides an option to initialize randomly, we use just this initialization). From the results in Figure 4.4a, we see perturbing the data before search reduces the cost reduction most effectively.

Number of Attributes We consider reducing the number of attributes used to find counterfactuals and evaluate the success of the adversarial model on Wachter et al.'s algorithm for the Communities and Crime dataset. Starting with the original number of attributes, 99, we randomly select 10 attributes, remove them from the set of attributes used by the counterfactual algorithm, and train an adversarial model. We repeat this process until we have 59 attributes left. We report the cost reduction due to $\boldsymbol{\delta}$ (Eq (4.9)) for each model, averaged over 5 runs. We observe that we are unable to find low cost recourses for adversarial model as we reduce the number of attributes, with minimal impact on accuracy (not in figure). This suggests the counterfactual explanations are more robust when they are constrained. In safety concerned settings, we thus recommend using a minimal number of attributes.

Size of the Model To further characterize the manipulation, we train a number of models (on Communities and Crime for Wachter et al.'s) that vary in their size. We show that as we increase the model size, we gain an even higher cost reduction, i.e. an 1.5× increase in the cost reduction when the similar additional parameters are added. This is not surprising, since more parameters provide further the flexibility to distort the decision surface as needed. As we reduce the size of the model, we see the opposite trend; the cost reduction reduces substantially when 4× fewer parameters are used. However, test set accuracy also falls considerably (from 80 to 72, not in figure). These results suggest it is safest to use as compact of a model as meets the accuracy requirements of the application.

Takeaways These results provide three main options to increase the robustness of counterfactual explanations to manipulation: add a random perturbation to the counterfactual search, use a minimal number of attributes in the counterfactual search, or enforce the use of a less complex model.

4.4 Potential Impacts

While we have shown that counterfactual explanations are vulnerable to manipulation, which is highly important for practitioners to be aware of, it is also critical to consider the potential impacts of developing adversarial attacks. In this section, we discuss potential impacts of developing adversarial models and evaluating on crime prediction tasks.

Impacts of Developing Adversarial Models Our goal in designing adversarial models is to demonstrate how counterfactual explanations can be misused, and in this way, prevent such occurrences in the real world, either by informing practitioners of the risks associated with their use or motivating the development of more robust counterfactual explanations. However, there are some risks that the proposed techniques could be applied to generate manipulative models that are used for harmful purposes. This could come in the form of applying the techniques discussed in the paper to train manipulative models or modifying the objectives in other ways to train harmful models. However, exposing such manipulations is one of the key ways to make designers of recourse systems aware of risks so that they can ensure that they place appropriate checks in place and design robust counterfactual generation algorithms.

Critiques of Crime Prediction Tasks In the chapter, we include the Communities and Crime data set. The goal of this data set is to predict whether violent crime occurs in communities. Using machine learning in the contexts of criminal justice and crime prediction has been extensively critiqued by the fairness community [Angwin et al., 2016, Rudin et al., 2020, Dressel and Farid, 2018]. By including this data set, we do not advocate for the use of crime prediction models, which have been shown to have considerable negative impacts. Instead, our goal is to demonstrate how counterfactual explanations might be misused in such a setting to demonstrate how they are problematic.

4.5 Discussion

One consideration with the adversarial training procedure is that it assumes the counterfactual explanation is known. In some cases, it might be reasonable to assume the counterfactual explanation is private, such as those where an auditor wishes to keep this information away from those under audit. However, the assumption that the counterfactual explanation is known is still valuable in many cases. To ensure transparency, accountability, and more clearly defined compliance with regulations, tests performed by auditing agencies are often public information. As one real-world example, the EPA in the USA publishes standard tests they perform [epa, 1994]. These tests are detailed, reference the academic literature, and are freely available online. Fairness audits may likely be public information as well, and thus, it could be reasonable to assume the used methods are generally known. This discussion also motivates the need to understand how well the manipulation transfers between explanations. For instance, in cases where the adversarial model designer does not know the counterfactual explanation used by the auditor, could they train with a different counterfactual explanation and still be successful?

Our results also motivate several futher research directions. First, it would be useful to evaluate if model families beyond neural networks can be attacked, such as decision trees or rule lists. In this work, we consider neural networks because they provide the capacity to optimize the objectives in Equations (4.1) and (4.2) as well as the (over) expressiveness necessary to make the attack successful. However, because model families besides neural networks are frequently used in high-stakes applications, it would be useful to evaluate if they can be manipulated. Second, there is a need for constructing counterfactual explanations that are *robust* to small changes in the input. Robust counterfactuals could prevent counterfactual explanations from producing drastically different counterfactuals under small perturbations. Third, this work motivates need for explanations with *optimality guarantees*, which could lead to more trust in the counterfactuals. Last, it could be useful to study when practitioners should use simpler models, such as in consequential domains, to have more knowledge about their decision boundaries, even if it is at the cost of accuracy.

4.6 Summary of Contribution

In this chapter, we demonstrate a critical vulnerability in counterfactual explanations and show that they can be manipulated, raising questions about their reliability. We show such manipulations are possible across a variety of commonly-used counterfactual explanations, including Wachter [Wachter et al., 2018], a sparse version of Wachter, Counterfactuals guided by prototypes [Van Looveren and Klaise, 2019], and DiCE [Mothilal et al., 2020]. These results bring into the question the trustworthiness of counterfactual explanations as a tool to recommend recourse to algorithm stakeholders. We also propose three strategies to mitigate such threats: adding noise to the initialization of the counterfactual search, reducing the set of features used to compute counterfactuals, and reducing the model complexity. This chapter is based on the publication:

- *Counterfactual Explanations Can Be Manipulated* (Slack et al. [2021b], NeurIPS 2021)

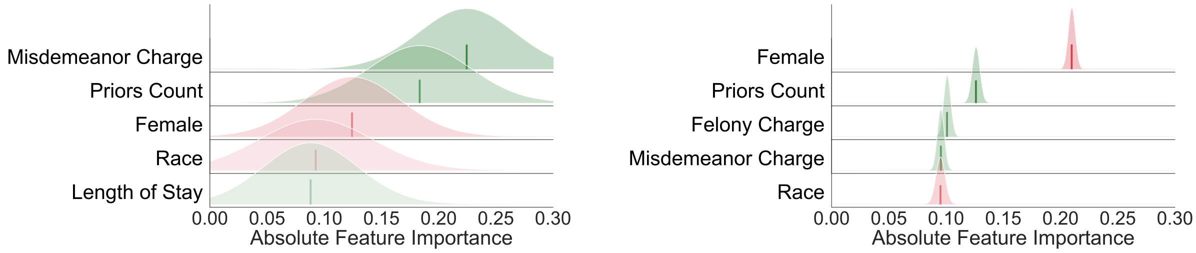
which has 49 citations at the time of writing.

This author of this dissertation is the primary author of this publication, and he developed the methods and produced the results.

Chapter 5

Reliable Local Post Hoc Explanations

Though local post hoc explanations are not robust in an adversarial setting, they also suffer from several fundamental issues in normal use. For instance, post hoc explanations may be unstable [Ghorbani et al., 2019, Slack et al., 2020, Dombrowski et al., 2019, Adebayo et al., 2018, Alvarez-Melis and Jaakkola, 2018], i.e., negligibly small perturbations to an instance can result in substantially different explanations. These methods are also inconsistent [Lee et al., 2019] i.e., multiple runs on the same input instance with the same parameter settings may result in vastly different explanations. There are also no reliable metrics to ascertain the quality of the explanations output by these methods. Commonly used metrics such as explanation fidelity rely heavily on the implementation details of the explanation method (e.g., the perturbation function used in LIME) and do not provide a true picture of the explanation quality [Tan et al., 2019]. Furthermore, there exists little to no guidance on determining the values of certain hyperparameters that are critical to the quality of the resulting local explanations (e.g., number of perturbations in case of LIME). Local explanation methods are also computationally inefficient i.e., they typically require a large number of black box model queries to construct local approximations [Chen et al., 2019b]. This can be prohibitively slow especially in case of complex neural models.



(a) Explanation computed with 100 perturbations (b) Explanation with 2000 perturbations

Figure 5.1: **Example explanations** on for an instance from the COMPAS dataset, where vertical lines indicate the feature importance by LIME (red is negative effect, green is positive) and the shaded region visualizes the uncertainty estimated by BayesLIME. While LIME produces very different and contradictory feature importance for different number of perturbations (5.1a and 5.1b), BayesLIME provides more context. The overlapping uncertainty intervals in the explanation computed with 100 perturbations (5.1a) indicate that it is unclear which feature is the most important. However, the tighter uncertainty intervals in the explanation computed with 2K perturbations (5.1b) clearly indicates that `Female` is the most important.

In this chapter, we identify that modeling uncertainty in black box explanations is the key to addressing all the aforementioned challenges. To this end, we propose a novel Bayesian framework for generating local explanations along with their associated uncertainty. We instantiate this framework to obtain Bayesian versions of LIME and KernelSHAP, namely BayesLIME and BayesSHAP, that not only output point-wise estimates of feature importance but also their associated uncertainty in the form of credible intervals (See Figure 5.1). We derive closed form expressions for the posteriors of the explanations thereby eliminating the need for any additional computational complexity. The credible intervals produced by our framework not only allow us to make concrete inferences about the quality of the resulting explanations but also produce explanations that satisfy user specified levels of uncertainty (e.g., an end user may request for explanations that satisfy a certain 95% confidence level). In addition, the resulting explanations are also highly consistent and stable. This contribution makes the first attempt at addressing several critical challenges in popular explanation methods in one-shots, thereby generating consistent, stable, and reliable explanations with guarantees in a computationally efficient manner.

We carry out theoretical analysis that leverages the measures of uncertainty (credible intervals) produced by our framework to estimate the values of critical hyperparameters. More specifically, we derive a closed form expression for the number of perturbations required to generate explanations that satisfy desired levels of confidence. We also propose a novel sampling technique called *focused sampling* that leverages uncertainty to determine how to sample perturbations for faster convergence, thereby enabling our framework to generate explanations in a computationally efficient manner.

We evaluate the efficacy of the proposed framework on a variety of datasets including COMPAS, German Credit, ImageNet, and MNIST. Our results demonstrate that the explanations generated by our framework are not only highly reliable, but also very consistent and stable (53% more stable than LIME/SHAP on an average). Our experimental results also confirm that we can accurately estimate the number of perturbations needed to generate explanations with a desired level of uncertainty, and that our uncertainty sampling technique speeds up the process of generating explanations by up to a factor of 2 relative to random sampling of perturbations. Lastly, we carry out a user study with 31 human subjects to evaluate the quality of the explanations generated by our framework, demonstrating that our explanations accurately capture the importance of the most influential features.

5.1 Our Framework: Bayesian Local Explanations

In this section, we introduce our Bayesian framework which is designed to capture the uncertainty associated with local explanations of black box models. First, we discuss the generative process and inference procedure for the framework. Then, we highlight how our framework can be instantiated to obtain Bayesian versions of LIME and SHAP. Lastly, we present detailed theoretical analysis for estimating the values of critical hyperparameters, and discuss how to efficiently construct highly accurate explanations with uncertainty guarantees using our framework.

5.1.1 Constructing Bayesian Local Explanations

Our goal here is to explain the behavior of a given black box model f in the vicinity of an instance x while also capturing the uncertainty associated with the explanation. To this end, we propose a Bayesian framework for constructing local linear model based explanations and capturing their associated uncertainty. We model the black box prediction of each perturbation z as a linear combination of the corresponding feature values ($\phi^T z$) plus an error term (ϵ) as shown in Eqn (5.2). While the weights of the linear combination ϕ capture the feature importances and thereby constitute our explanation, ϵ captures the error that arises due to the mismatch between our explanation ϕ and the local decision surface of the black box model f. Our complete generative process is shown below:

$$y|z, \phi, \epsilon \sim \phi^T z + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \frac{\sigma^2}{\pi_x(z)}) \tag{5.1}$$

$$\phi|\sigma^2 \sim \mathcal{N}(0, \sigma^2 \mathbb{I}) \qquad \sigma^2 \sim \text{Inv-}\chi^2(n_0, \sigma_0^2). \tag{5.2}$$

The error term is modeled as a Gaussian whose variance relies on the proximity function $\pi_x(z)$ i.e., $\epsilon \sim \mathcal{N}(0, \frac{\sigma^2}{\pi_x(z)})$. This proximity function ensures that perturbations closer to the data point x are modeled accurately, while allowing more room for error in case of perturbations that are farther away. $\pi_x(z)$ can be computed using cosine or l_2 distance or other game theoretic principles similar to that of LIME and KernelSHAP. The conjugate priors on ϕ and σ^2 are shown in Eqn (5.2). Note that, the distributions on error ϵ and feature importance ϕ are both dependent on the parameter σ^2. The fact that the prior on the feature importances considers σ^2 has an intuitive interpretation: if we have prior knowledge that the error of the explanation is small, we expect to be more confident about the feature importances.

Similarly, if we have prior knowledge the error is large, we expect to be less confident about the feature importances.

Thus, our generative process corresponds to the Bayesian version of the weighted least squares formulation of LIME and KernelSHAP outlined in Equation. (2.3), with additional terms to model uncertainty. As in Eqns. (5.2), the process captures two sources of uncertainty in local explanations: 1) ***feature importance uncertainty***: the uncertainty associated with the feature importances ϕ, and (2) ***error uncertainty***: the uncertainty associated with the error term ϵ which captures how well our explanation ϕ models the local decision surface of the underlying black box.

Inference Our inference process involves estimating the values of two key parameters: ϕ and σ^2. By doing so, we can compute the local explanation as well as the uncertainties associated with feature importances and the error term. Posterior distributions on ϕ and σ^2 are normal and scaled Inv-χ^2, respectively, due to the corresponding conjugate priors [Moore, 1995]:

$$\begin{aligned}
\sigma^2|\mathcal{Z}, Y &\sim \text{Scaled-Inv-}\chi^2\left(n_0 + N, \frac{n_0\sigma_0^2 + Ns^2}{n_0 + N}\right) \\
\phi|\sigma^2, \mathcal{Z}, Y &\sim \text{Normal}(\hat{\phi}, V_\phi\sigma^2) \qquad (5.3)
\end{aligned}$$

Further, $\hat{\phi}$, V_ϕ, and s^2 can be directly computed:

$$\begin{aligned}
\hat{\phi} &= V_\phi(\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))Y) \\
V_\phi &= \left(\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + \mathbb{I}\right)^{-1} \qquad (5.4)
\end{aligned}$$

$$s^2 = \frac{1}{N}\left[(Y - \mathcal{Z}\hat{\phi})^T \text{diag}(\Pi_x(\mathcal{Z}))(Y - \mathcal{Z}\hat{\phi}) + \hat{\phi}^T\hat{\phi}\right] \qquad (5.5)$$

Details of the complete inference procedure including derivations of Eqns. (5.3-5.5) are provided in the Appendix A.1. Note that our estimate of the posterior mean feature importances $\hat{\phi}$ (Eqn. (5.4)) is the same as that of the feature importances computed in the cases of LIME and KernelSHAP (Eqn. (2.3)).

Remark 5.1. *If we use the same proximity function* $\pi_x(z)$ *in our framework as in LIME or KernelSHAP, the posterior mean of the feature importance* $\hat{\phi}$ *output by our framework (Eq* (5.4)*) will be equivalent to the feature importances output by LIME or KernelSHAP, respectively.*

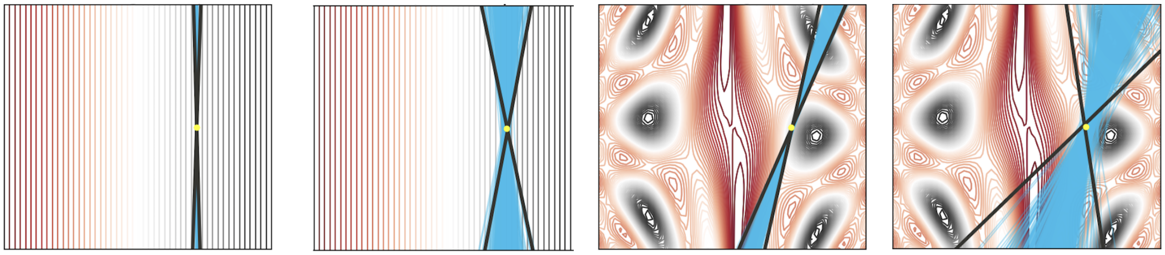
Feature Importance Uncertainty To obtain the local feature importances and their associated uncertainty, we first compute the posterior mean of the local feature importances $\hat{\phi}$ using the closed form expression in Eqn. (5.5). We then estimate the credible interval (measure of uncertainty) around the mean feature importances by repeatedly sampling from the posterior distribution of ϕ (Eq (5.3)).

Error Uncertainty The error term ϵ can serve as a proxy for explanation quality because it captures the mismatch between the constructed explanation and the local decision surface of the underlying black box. We first calculate the marginal posterior distribution of ϵ by leveraging Eqn (5.2) and integrating out σ^2. This results in a three parameter Student's t distribution:

$$\epsilon|\mathcal{Z}, Y \sim t_{(\mathcal{V}=n_0+N)}(0, \frac{n_0\sigma_0^2 + Ns^2}{n_0 + N}). \tag{5.6}$$

We then evaluate the probability density function (PDF) of the above posterior at 0, i.e., $P(\epsilon = 0)$ by substituting the value of s^2 computed using Eqn. (5.5) into the Student's t distribution above (Eqn. (5.6)). The resulting expression gives us the probability density that the explanation output by our framework perfectly captures the local decision surface underlying the black box. This operation is performed in constant time, adding minimal overhead to non-Bayesian LIME and SHAP. We illustrate how these computed intervals capture the variance in the explanations in Figure 5.2.

Proposition 5.1. *As the number of perturbations around* x *goes to* ∞ *i.e.,* $N \to \infty$*: (1) the estimate of* ϕ *converges to the true feature importance scores, and its uncertainty to* 0*. (2)*



(a) linear, many samples (b) linear, fewer samples (c) nonlinear, many samples (d) nonlinear, fewer samples

Figure 5.2: Rerunning LIME local explanations 1000 times and BayesLIME *once* for linear and non-linear toy surfaces using few (25) and many (250) perturbations. The linear surface is given as $p(y) \propto x_1$ and the non linear surface is defined as $p(y) \propto \sin(x_1/2) * 10 + \cos(10 + (x_1 * x_2)/2) * \cos(x_1)$. We plot each run of LIME in blue and the BayesLIME 95% credible region of the feature importance ϕ in black. We see that LIME variance is higher with fewer samples and a less linear surface. BayesLIME captures the relative difficulty of explaining each surface through the width the credible region. For instance, BayesLIME is most uncertain in the nonlinear, few samples case because this surface is the most difficult to explain.

uncertainty of the error term ϵ converges to the bias of the local linear model ϕ. [Details in Appendix A.2]

BayesLIME and BayesSHAP Our framework can be instantiated to obtain the Bayesian version of LIME by setting the proximity function to $\pi_x(z) = \exp(-D(x,z)^2/\sigma^2)$ where D is a distance metric (e.g. cosine or l_2 distance), and n_0 and σ_0^2 to small values (10^{-6}) so that the prior is uninformative. We compute feature importance uncertainty and error uncertainty for LIME's feature importances.

Our framework can also be instantiated to obtain the Bayesian version of KernelSHAP by setting uninformative prior on σ^2 and $\pi_x(z) = \frac{d-1}{(d \text{ choose } |z|)|z|(d-|z|)}$ where $|z|$ denotes the number of the variables in the variable combination represented by the data point z i.e., the number of non-zero valued features in the vector representation of z. Note that the original SHAP method views the problem of constructing a local linear model as estimating the Shapley values corresponding to each of the features [Lundberg and Lee, 2017b]. These Shapley values represent the contribution of each of the features to the black box prediction i.e.,

$f(x) = \phi_0 + \sum \phi_i$. Therefore, the measures of uncertainty output by our method BayesSHAP capture the reliability of the estimated variable contributions.

To encourage BayesLIME and BayesSHAP explanations to be sparse, we can use dimensionality reduction or feature selection techniques as used by LIME and SHAP to obtain the top K features [Ribeiro et al., 2016b, Lundberg and Lee, 2017b, Sokol et al., 2019]. We can then construct our explanations using the data corresponding to these top K features.

5.1.2 Estimating the Number of Perturbations

One of the major drawbacks of approaches such as LIME and KernelSHAP is that they do not provide any guidance on how to choose the number of perturbations, a key factor in obtaining reliable explanations in an efficient manner. To address this, we leverage the uncertainty estimates output by our framework to compute *perturbations-to-go* (G), an estimate of how many *more* perturbations are required to obtain explanations that satisfy a desired level of certainty. This estimate thus *predicts* the computational cost of generating an explanation with a desired level of certainty and can help determine whether it is even worthwhile to do so. The user specifies the confidence level of the credible interval (denoted as α) and the *maximum* width of the credible interval (W), e.g. "width of 95% credible interval should be less than 0.1" corresponds to $\alpha = 0.95$ and $W = 0.1$. To estimate G for the local explanation of a data point x, we first generate S perturbations around x (where S is small and chosen by the user) and fit a local linear model using our method[1]. This provides initial estimates of various parameters shown in Eqns (5.3)-(5.5) which can then be used to compute G.

Theorem 5.1. *Given S seed perturbations, the number of additional perturbations required (G) to achieve a credible interval width W of feature importance for a data point x at*

---

[1] We assume a simplified feature space where features are present or absent according to Bernoulli(.5). As in Ribeiro et al. [2016b], these *interpretable* features are flexible and can encode what is important to the end user.

user-specified confidence level α can be computed as:

$$G(W, \alpha, x) = \frac{4s_S^2}{\bar{\pi}_S \times \left[\frac{W}{\Phi^{-1}(\alpha)}\right]^2} - S \tag{5.7}$$

where $\bar{\pi}_S$ is the average proximity $\pi_x(z)$ for the S perturbations, s_S^2 is the empirical sum of squared errors (SSE) between the black box and local linear model predictions, weighted by $\pi_x(z)$, as in (5.5)*, and $\Phi^{-1}(\alpha)$ is the two-tailed inverse normal CDF at confidence level α.*

Proof: In this derivation, the perturbation matrices $\mathcal{Z}$ have elements $\mathcal{Z}_{ij} \in \{0, 1\}$ where each $\mathcal{Z}_{ij} \sim$ Bernoulli(0.5). Note, in these proofs, we take take the priors to be set as in BayesLIME and BayesSHAP, i.e., they have hyperparameter values close to 0. Also, we use N to denote the *total* perturbations while S denotes the perturabtions collected *so far.* We use three assumptions stated as follows. First, $\frac{\bar{\pi}N}{2}$ is sufficiently large such at $\frac{\bar{\pi}N}{2} + 1$ is equivalent to $\frac{\bar{\pi}N}{2}$. Second, N is sufficiently large such that $N + 1$ is equivalent to N and $\frac{N}{N-2}$ is equivalent to 1. Third, the product of $\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z}$ within V_ϕ can be taken at its expected value. First, we state the marginal distribution over feature importance ϕ_i where i is an arbitrary feature importance $i \in d$. This given as

$$\phi_i | \mathcal{Z}, Y \sim t_{\mathcal{V}=N}(\hat{\phi_i}, V_{\phi_{ii}} s^2) \tag{5.8}$$

where $V_\phi = (\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}$. Recall each $\mathcal{Z}_{ij}$ is given $\sim$ Bern(.5) we use the third assumption to write V_ϕ is $\frac{\bar{\pi}N}{2} + 1$ for the on diagonal elements and $\frac{\bar{\pi}N}{4}$ for the off diagonal elements. We can see this is the case considering that each element in $\mathcal{Z}$ is a Bern(.5) draw. We drop the $1's$ due to the first assumption.

Let $k = \frac{\bar{\pi}N}{2}$. It follows directly from the Sherman Morrison formula that the i-th and j-th entries of V_ϕ are given as [Golub and Van Loan, 1996],

$$(V_\phi)_{ij} = \begin{cases} \frac{2}{k} - \frac{2}{k(N+1)} & i = j \\ -\frac{2}{k(N+1)} & i \neq j \end{cases} \qquad (V_\phi)_{ii} = \frac{4}{\bar{\pi}(N+1)} \tag{5.9}$$

We see that the diagonals are the same. Thus, we take the perturbations-to-go (PTG) estimate in terms of a single marginal ϕ_i. Substituting in the s^2 estimate s_S^2 and using the second assumption, we write the variance of marginal ϕ_i as

$$\mathrm{Var}(\phi_i) = \frac{4s_S^2}{\bar{\pi}(N+1)} \frac{N}{N-2} \tag{5.10}$$

$$= \frac{4s_S^2}{\bar{\pi} \times N} = \frac{4s_S^2}{\bar{\pi} \times \mathrm{Var}(\phi_i)} \tag{5.11}$$

Because feature importance uncertainty is in the form of a credible interval, we use the normal approximation of $\mathrm{Var}(\phi_i)$ and write

$$N = \frac{4s_S^2}{\bar{\pi} \times \left[\frac{W}{\Phi^{-1}(\alpha)}\right]^2} \tag{5.12}$$

where W is the desired width, α is the desired confidence level, and $\Phi^{-1}(\alpha)$ is the two-tailed inverse normal CDF. Finally, we subtract the initial S samples. □

5.1.3 Focused Sampling of Perturbations

Perturbations-to-go (G) provides us with an estimate of how many samples are required to achieve reliable explanations. However, if G is large, querying the black-box model for its predictions on a large number of perturbations can be computationally expensive for larger models [Denton et al., 2014, Jaderberg et al., 2014]. To reduce this cost, we develop an alternative sampling procedure called *focused sampling* which leverages uncertainty estimates to query the black box in a more targeted fashion (instead of querying randomly), thereby reducing the computational cost associated with generating reliable explanations. Inspired by active learning [Settles, 2010], focused sampling strategically prioritizes perturbations whose predictions the explanation is most uncertain about, when querying the black box. This enables the focused sampling procedure to query the black box only for the predictions of the most informative perturbations and thereby learn an accurate explanation with far fewer queries to the black box.

To determine how uncertain our explanation ϕ is about the black box label for any given instance z, we first compute the posterior predictive distribution for z (derivation in Appendix A.1), given as $\hat{y}(z)|\mathcal{Z}, Y \sim t_{(\mathcal{V}=N)}(\hat{\phi}^T z, (z^T V_\phi z + 1)s^2)$. The variance of this three parameter student's t distribution is,

$$\mathrm{var}\left(\hat{y}(z)\right) = ((z^T V_\phi z + 1)s^2)(N/(N-2)) \tag{5.13}$$

We refer to this variance as the *predictive variance* $\mathrm{var}(\hat{y}(z))$, and it captures how uncertain our explanation ϕ is about the black box prediction.

The focused sampling procedure first fits the explanation with an initial S perturbations (where S is a small number). We then iterate the following procedure until the desired explanation certainty level is reached. We draw a batch of A candidate perturbations, compute their predictive variance with the Bayesian explanation, and induce a distribution over the perturbations by running softmax on the variances with tempurature parameter τ. We draw a batch of B perturbations from this distribution and query the black box model for their labels. Finally, we refit the Bayesian explanation on all the labeled perturbations collected so far. We provide pseudocode for the uncertainty sampling procedure in Algorithm 1.

Algorithm 1 Focused sampling for local explanations

```
Require: Model $f$, Data instance $x$, Number of perturbations $N$, Number of seed perturbations $S$, Batch size $B$, Pool size $A$, tempurature $\tau$
function FOCUSED SAMPLE
    Initialize $\mathcal{Z}$ with $S$ seed perturbations.
    Fit $\hat{\phi}$ on $\mathcal{Z}$                                    ▷ Using Eqn (5.4)
    for $i \leftarrow 1$ to $N - S$ in increments of $B$ do
        $\mathcal{Q} \leftarrow$ Generate $A$ candidate perturbations
        Compute $\mathrm{var}(\hat{y}(z))$ on $\mathcal{Q}$              ▷ Using Eqn (5.13)
        Define $\mathcal{Q}_{\text{dist}}$ as $\propto \exp(\mathrm{var}(\hat{y}(z))/\tau)$
        $\mathcal{Q}_{\text{new}} \leftarrow$ Draw $B$ samples from $\mathcal{Q}_{\text{dist}}$
        $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathcal{Q}_{\text{new}}$; Fit $\hat{\phi}$ on $\mathcal{Z}$     ▷ Using Eqn (5.4)
    end for
    return $\hat{\phi}$
end function
```

5.2 Experiments

We evaluate the proposed framework by first analyzing the quality of our uncertainty estimates i.e., feature importance uncertainty and error uncertainty. We also assess our estimates of required perturbations (G), and evaluate the computational efficiency of focused sampling. Last, we describe a user study with 31 subjects to assess the informativeness of the explanations output by our framework.

Setup We experiment with a variety of real world datasets spanning multiple applications (e.g., criminal justice, credit scoring) as well as modalities (e.g., structured data, images). Our first structured dataset is **COMPAS** [Angwin et al., 2016], containing criminal history, jail and prison time, and demographic attributes of 6172 defendants, with class labels that represent whether each defendant was rearrested within 2 years of release. The second structured dataset is the **German Credit** dataset from the UCI repository [Dua and Graff, 2017] containing financial and demographic information (including account information, credit history, employment, gender) for 1000 loan applications, each labeled as a "good" or "bad" customer. We create 80/20 train/test splits for these two datasets, and train a random forest

	BayesLIME	**BayesSHAP**		**BayesLIME**	**BayesSHAP**
TABULAR DATASETS			MNIST		
COMPAS	95.5	87.9	Digit 1	95.8	98.4
German Credit	96.9	89.6	Digit 2	95.8	97.4
IMAGENET			Digit 3	95.2	96.3
Corn	94.6	91.8	Digit 4	97.2	90.1
Broccoli	91.4	89.2	Digit 5	95.2	95.6
French Bulldog	94.8	89.9	Digit 6	96.7	96.8
Scuba Diver	92.4	94.6	Digit 7	95.7	95.3

Table 5.1: **Evaluating Credible Intervals.** We report the % of time the 95% credible intervals with 100 perturbations include their true values (estimated on $10,000$ perturbations). Closer to 95.0 is better. Both BayesLIME and BayesSHAP are well calibrated.

classifier (sklearn implementation with 100 estimators) as *black box* models for each (test accuracy of 82.8% and 72.5%, respectively). We also include popular image datasets–MNIST and Imagenet. For the **MNIST** [LeCun et al., 2010] handwritten digits dataset, we train a 2-layer CNN to predict the digits (test accuracy of 99.2%). For **Imagenet** [Deng et al., 2009], we use the off-the-shelf VGG16 model [Simonyan and Zisserman, 2015] as the black box. We select a sample of 100 images of the following classes French Bulldog, Scuba Diver, Corn, and Broccoli to use in the experiments. For generating explanations, we use standard implementations of the baselines LIME and KernelSHAP with default settings [Ribeiro et al., 2016b, Lundberg and Lee, 2017b]. For images, we construct super pixels as described in [Ribeiro et al., 2016b] and use them as features (number of super pixels is fixed to 20 per image). For our framework, the desired level of certainty is expressed as the width of the 95% credible interval.

Quality of Uncertainty Estimates A critical component of our explanations is the feature importance uncertainty. To evaluate the correctness of these estimates, we compute how often *true* feature importances lie within the 95% credible intervals estimated by BayesLIME and BayesSHAP. Note, that by *true* feature importance, we refer to the best fit linear model output using either the LIME or SHAP kernels. We evaluate the quality of our

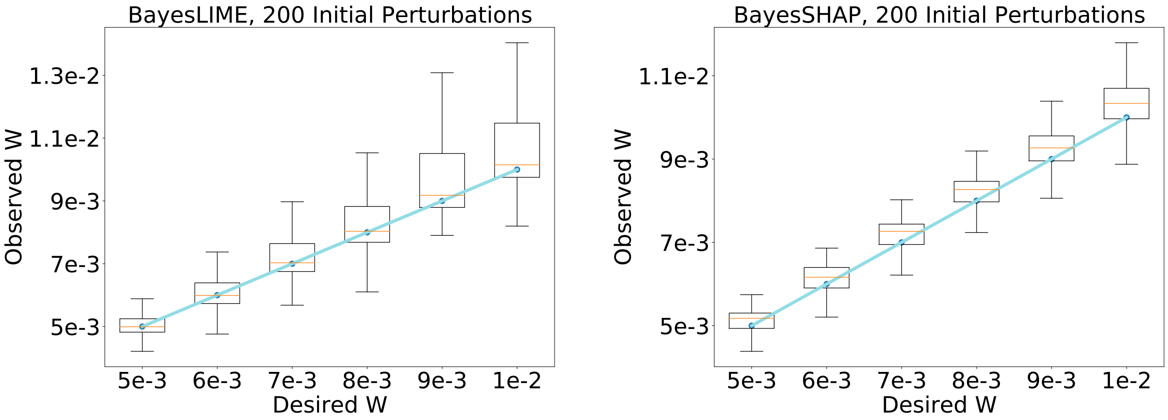


Figure 5.3: **Perturbations-to-go (G).** We generate explanation with G perturbations, where G is computed using the *desired* credible interval width (x-axis), and compare desired levels to the *observed* credible interval width (y-axis) (blue line indicates ideal calibration). Results are averaged over 100 MNIST images of the digit "4" We see that G provides a good approximation of the additional perturbations needed.

credible interval estimates by running our methods with 100 perturbations to estimate feature importances and taking the corresponding 95% credible intervals for each test instance. We compute what fraction of the true feature importances fall within our 95% credible intervals. Note, because there are no methods to provide uncertainty estimates for LIME and SHAP, we do not provide further baselines. Since we do not have access to the true feature importances of the complex black box models, following Prop 5.1, we use feature importances computed using a large value of N ($N = 10,000$), and treat the resulting estimates as ground truth.

Results for BayesLIME in Table 5.1 indicate that the true feature importances are close to ideal and indicate the estimates are well calibrated. While the estimates by BayesSHAP are somewhat less calibrated (true feature importances fall within our estimated 95% credible intervals about 89.2 to 98.4% of the time), they still are quite close to ideal. All in all, these results confirm that the credible intervals learned by our methods are well calibrated and therefore highly reliable in capturing the uncertainty of the feature importances.

Correctness of Estimated Number of Perturbations We assess whether our estimate of *perturbations-to-go* (G; Section 5.1.2) is an accurate estimate of the *additional* number of perturbations needed to reach a desired level of feature importance certainty. We carry out this experiment on MNIST data for the digit “4” and use $S = 200$ as the initial number of perturbations to obtain a preliminary explanation and its associated uncertainty estimates. We then leverage these estimates to compute G for 6 different certainty levels. First, we observe significant differences in G estimates across instances, i.e. number of perturbations needed to obtain a particular level of certainty varied significantly across instances–ranging from 200-$5,000$ for the lowest level of certainty to 200-$20,000$ for higher levels of certainty. Next, for each image and certainty level, we run our method for the estimated number of perturbations (G) to determine if the observed estimates of uncertainty (observed credible interval width W) match the desired levels of uncertainty (desired credible interval width W). Results in Figure 5.3 show that the observed and desired levels of certainty are well calibrated, demonstrating that G estimates are reliable approximations of the additional number of perturbations needed.

Efficiency of Focused Sampling *Focused sampling* uses the *predictive variance* to strategically choose perturbations that will reduce uncertainty in order to be labeled by the black box (section 5.1.3). Here, we will evaluate the efficiency of the focused sampling procedure. First, we assess whether focused sampling converges (as measured by error uncertainty ($P(\epsilon = 0)$)) more efficiently than random sampling. To this end, we experiment with BayesLIME on Imagenet data for the “French bulldog” class to carry out this analysis. This setting replicates scenarios where LIME is applied to a computationally expensive black box model, making it highly desirable to limit the number of perturbations to reduce total running time. We run each sampling strategy for 2,000 perturbations and plot the number of model queries versus error uncertainty. During focused sampling, we set the batch size B to 50. The results in Figure 5.4 show that focused sampling results in faster convergence to reliable and high quality explanations; focused sampling stabilizes within a couple hundred model

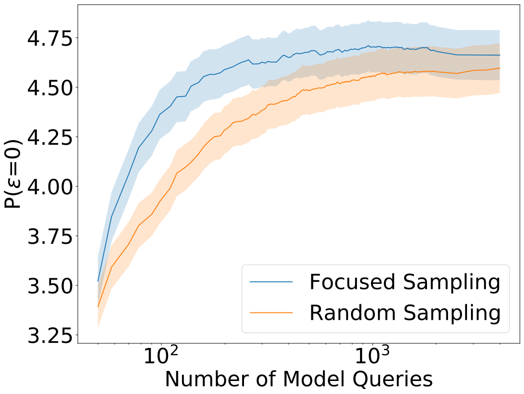


Figure 5.4: **Efficiency of focused sampling** for 100 Imagenet "French bulldog" images, with random sampling as a baseline. We provide mean and standard error. We assess the efficiency of focused sampling by comparing *error uncertainty* over model queries and show quicker convergence than random sampling.

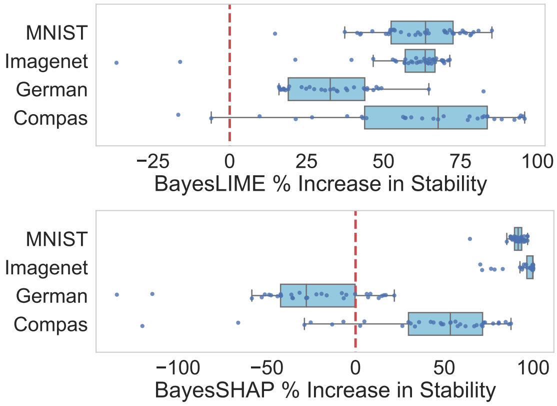


Figure 5.5: **Assessing the % increase in stability** of BayesLIME and BayesSHAP over LIME and SHAP respectively. Our Bayesian methods are significant more stable ($\rho <$ 1e-2 according to Wilcoxon signed-rank test) except for BayesSHAP on German Credit, where there is not a significant difference between the methods ($\rho > 0.05$).

queries while random sampling takes over 1,000. Note, as the inefficiency of querying the black box model increases, the advantages of focused sampling decreasing total running time of the explanations will only become more pronounced. These results clearly demonstrate that focused sampling can significantly speed up the process of generating high quality local explanations.

Stability of BayesLIME & BayesSHAP Recall that LIME & SHAP are not stable: small changes to instances can produce substantially different explanations. We consider whether BayesLIME & BayesSHAP produce more stable explanations than their LIME & SHAP counterparts. To perform this analysis, we use the local Lipschitz metric for explanation stability [Alvarez-Melis and Jaakkola, 2018]:

$$\hat{L}(x_i) = \underset{x_j \in N_\epsilon(x_i)}{\text{argmax}} \frac{||\phi_i - \phi_j||_2}{||x_i - x_j||_2} \tag{5.14}$$

where x_i refers to an instance, $N_\epsilon(x_i)$ is the ϵ-ball centered at x_i, and ϕ_i and ϕ_j are the explanation parameters for x_i and x_j. Lower values indicate more stable explanations. We follow the setup outline by Alvarez-Melis and Jaakkola [2018] and compute the local Lipschitz values, comparing both LIME & BayesLIME and SHAP & BayesSHAP across Compas, German Credit, MNIST digit "4", and Imagenet "French Bulldog." We perform the comparison using the default number of perturbations in both LIME & SHAP, and use this same number in the respective Bayesian variants and set the batch size B to half this value. We use focused sampling for BayesLIME and BayesSHAP, and report the % increase in stability of these approaches over LIME and SHAP for 40 test points. The results given in Figure 5.5 show a clear improvement (on average 53%) in stability in all cases except German Credit for BayesSHAP. Further, we run a Wilcoxon signed-rank test and find our results are statistically significant in all cases ($\rho <$ 1e-2) except for BayesSHAP for German Credit, where there is not a significant difference between the methods ($\rho > 0.05$). These results demonstrate BayesLIME and BayesSHAP are more stable than previous methods.

User Study We perform a user study with 31 subjects to compare BayesLIME and LIME explanations on MNIST. We evaluate the following: are explanations with low levels of uncertainty (i.e., most confident explanations) more meaningful to humans? To answer this question, we follow prior work and mask the most important features selected by BayesLIME and LIME [Schwab and Karlen, 2019, Lundberg and Lee, 2017b]. We ask users to guess the digit of the masked images. The better the explanation, the more difficult it should be for the users to get it right. Further, the choice to mask the important features is motivated by its success in prior work. We randomly select 15 correctly predicted test images, generate explanations by sweeping over a range of perturbation amounts $[10^{-5}, ..., 10^{3.5}]$ incremented by 0.5. We choose the *top* explanation for each image based on either fidelity (for LIME) or $P(\epsilon = 0)$ (for BayesLIME). We sent the user study out to students and researchers with background in computer science. We find that the explanations output by our methods focus on more informative parts of the image, since hiding them makes it difficult for humans to guess the digit. Users had an error rate of 25.7% for LIME, while it was 30.7% for BayesLIME, both with standard error 0.003 ($\rho = 0.028$ through a one-tailed two sample t-test). This result indicates that our method BayesLIME and the associated measure of explanation uncertainty result in more high quality and reliable explanations compared to LIME and its associated fidelity metric.

5.3 Discussion

While the Bayesian framework addresses several critical challenges (i.e., consistency, stability, modeling uncertainty) associated with LIME and SHAP, there are still certain aspects where it would exhibit the same shortcomings as LIME and SHAP [Lundberg and Lee, 2017b, Agarwal et al., 2021b]. For instance, if the local decision surface of a given black box classifier is highly non-linear, our framework, which relies on local linear approximations, may not be able to capture this non-linear decision surface accurately. In addition, if the perturbation sampling procedures used in LIME and SHAP are used in BayesLIME and BayesSHAP, they will likely be vulnerable to the attacks proposed by Slack et al. [2020]. In the future, it would be interesting to extend our framework to produce global explanations with uncertainty guarantees and explore how uncertainty quantification can help calibrate user trust in model explanations.

5.4 Summary of Contributions

In this chapter, we developed a Bayesian framework for generating local explanations along with their associated uncertainty. We instantiated this framework to obtain Bayesian versions of LIME and SHAP that output pointwise estimates of feature importances as well as their associated credible intervals. These intervals enabled us to infer the quality of the explanations and output explanations that satisfied user specified levels of uncertainty. We carried out theoretical analysis that leverages these uncertainty measures (credible intervals) to estimate the values of critical hyperparameters (e.g., the number of perturbations). We also proposed a novel sampling technique called focused sampling that leverages uncertainty estimates to determine how to sample perturbations for faster convergence. The text in this chapter is based on the following publication:

- *Reliable Post hoc Explanations: Modeling Uncertainty in Explainability* (Slack et al. [2021a], NeurIPS 2021)

which has 66 citations at the time of writing.

The author of this dissertation is the primary author of the publication, and is responsible for the results.

Part II

Natural Language For Explaining & Developing Machine Learning Models

Chapter 6

Natural Language Conversations For Explainability With TalkToModel

In the first part of this dissertation, we showed how explanations are not robust and introduced new techniques to help overcome these issues. However, it remains an open question *how* end-users, such as domain experts or laypeople, are satisfied with current approaches to generating explanations, and—if not—(1) what are the fundamental limitations of the current explanations, (2) what are desirable approaches to explaining model algorithms in the real world, and (3) can we instantiate these approaches for successful results with end-users?

In this chapter, we fill this gap by retrospectively evaluating the existing approaches to explanability widely adopted by the research community through the lens of real-world decision-makers. In particular, we conducted interviews with domain experts in healthcare and policy-making to understand how they use explanations in their day-to-day work, the pain points they experience with existing explanations, and what would they like to see in the next generation of explanations. Building on the user study, we argue that *natural language conversations* are a promising avenue for future work for explainability.

Inspired by these findings, we introduce TalkToModel, a system that enables open-ended natural language conversations for explainability. To construct TalkToModel, we design a flexible domain specific language (DSL) that includes compositional operations for explainability and model analysis. We introduce methods for training LLMs to parse natural language into this DSL, which the system executes and the results are included into the system's conversational responses. In our evaluation, we find TalkToModel understands users with a high degree of accuracy and the DSL supports the vast majority of user queries. In addition, users strongly prefer TalkToModel over existing explainability systems, demonstrating the utility of natural language conversations to support model understanding.

6.1 User Study

Here, we discuss the study that we carried out with practitioners from healthcare and policy to understand how they use explanations in their day-to-day work, the pain points they experience with existing explanations, and what would they like to see in the next generation of explanations. More specifically, we conducted 30-minute long semi-structured interviews with 26 practitioners who regularly employ explainability techniques in their workflow. 14 out of these 26 (53.8%) practitioners are both medical doctors and researchers who actively use explanation methods to understand ML models that diagnose different kinds of diseases ranging from diabetes to rare cancers. The remaining 12 (46.2%) of these practitioners are policy researchers who are utilizing explanation methods to understand financial decision making (e.g., loan approvals) models. Furthermore, 18 out of 26 (69.2%) practitioners are male and the remaining 8 (30.7%) are female. 16 practitioners (61.5%) had more than a year of experience working with explainability tools, and the remaining 10 of them (38.5%) had about 6 months to a year of experience. All the practitioners in our study have used local post hoc explanation methods such as LIME and SHAP in their workflow, and 12 of them

(46.2%) also used various gradient based methods (e.g., GradCAM, Integrated Gradients etc.). 11 participants (42.3%) also mentioned that they *understand* the technical details of LIME, but none of the participants had any understanding of the inner workings and details of any other explanation methods.

6.1.1 Format

We began the interview by asking each of the participants about how exactly they leverage model explanations. All the participants said that they look at feature attributions output by post hoc explanation methods for each model prediction of interest, and that they specifically focus on the top 5 to 8 features that are driving the prediction. 21 out of 26 participants mentioned that they also look at the sign (or direction of the contribution) of the feature attribution for certain features of interest—e.g., is salary contributing positively to the loan approval decision? Lastly, 19 out of 26 (73.1%) participants mentioned that they also compare features w.r.t. their sign, rank, and feature importance values. Our interviews further included, but were not limited to the following questions:

- What do you like about model explanations output by state-of-the-art methods?
- What do you dislike about model explanations output by state-of-the-art methods?
- What other features should explanations have for you to comfortably use them in your day-to-day work? (24 out of 26 participants wanted some form of an interactive dialogue for explanations.)
- Would you prefer a one-shot (single) explanation or interactive dialogue style explanations? (We asked this question only if participants did not bring up interactive dialogue on their own; only 2 out of 26 participants did not)

- What are the key desiderata you would like to have in interactive dialog style explanations? (We asked this question to the 25 out of 26 participants who wanted interactive dialogue style explanations).

6.1.2 Results

Overall, while respondents were satisfied with many features of current explainability techniques, they pointed to several critical shortcomings with current methods. Further, respondents expressed a strong desire for interactive explanations and felt that natural language dialogues could serve as an advantageous type of interactive explanations. Last, interviewees felt that natural language dialogues could create a better explainability experience and identified critical criteria explainability dialogues should satisfy.

The Need for Interactive Explanations

During the interviews, respondents indicated several aspects of current explainability techniques they liked. Respondents most enjoyed getting some understanding of deep learning models (26/26 liked) and understanding which features contribute positively and negatively (21/26 liked). Slightly fewer respondents enjoyed seeing the essential features for predictions (19/26 liked) and comparing the relative importance of features (18/26 liked). All in all, respondents expressed that current explainability techniques help understand how machine learning models work and how different features affect the model predictions.

While respondents indicated they enjoyed certain features of explainability techniques, they also expressed several unsatisfactory aspects of explanations. Respondents were most dissatisfied with the lack of additional interaction with explanations after generation. Respondents answered that they were highly dissatisfied with the fact that conversations with the explanations are not possible (25/26 disliked), there is no capacity to follow up on explanations (24/26 disliked) interactively, nor ask custom questions (23/26 disliked). One respondent stated, “It is extremely frustrating to just look at one explanation [per prediction] and not be able to follow up on it!” Another indicated, “I should be able to ask custom questions [to the explanation] and get answers.” Respondents also disliked that they could not understand the accuracy of explanations (24/26 disliked). One of the interviewees described, “I don’t know anything about how correct the explanation is! How do you expect me to use it meaningfully? I constantly struggle with worrying about using an incorrect explanation and missing out on not using a correct explanation that is giving me more insights.” Slightly fewer respondents indicated they disliked the limited capacity of explanations to generate subgroup-level explanations (21/26 disliked). One respondent questioned, “Why is all explanation work focused on local explanations? I would like to see at least subgroup level explanations. I think there is one algorithm (MUSE?) but need a lot more work.” Overall, respondents expressed evident dissatisfaction with the one-off nature of explanations. Respondents felt that, in almost all cases, they had further follow-up questions for explanations to do with the explanation’s accuracy or additional tasks they would like the explanation to solve. The interviewees felt that the lack of interactivity with explanations is a significant shortcoming of existing techniques.

When we asked respondents what could improve explanations, respondents discussed several potential improvements. Overall, respondents expressed the strongest desire for explainability through fully-fledged conversations with ML models (25/26 said this was important). Multiple respondents voiced support for conversational explanations. One stated, “I can see myself using explainable tools a ton more if only it were like a free-flowing dialogue. Oh I can’t wait for that day.” Another said, “dialogue-based explanations will totally revolutionize how medical science uses ML. Wow, I am excited just thinking about the possibility.” Respondents also indicated the inclusion of reliable accuracy metrics for explanations as a critical place of improvement (24/26 said this was important). Slightly fewer respondents indicated that custom questions are vital for improving explanations (22/26 said this was important) and improving subgroup level explanations (24/26 said this was important). While the interviewees indicated several places to improve explanations (such as accuracy metrics), respondents most heavily fixated on the potential of having conversations with ML models to support explainability. Considering that respondents indicated a strong desire for dialogue-based explanation systems, we next discussed key desiderata for an explainability dialogue system.

6.1.3 Explainability Dialogue Desiderata from Interviewees

Respondents felt that fully-fledged conversations in natural language with explanations would help them better understand ML models. In addition, they felt that a explainability dialogue system could greatly help their explainability workflows. Further, respondents had numerous ideas about the system's capabilities, different ways the system could augment their explainability workflows, and what they hoped to get out of such a system.

Critically, many respondents envisioned explainability dialogues happening much like conversations with colleagues where the goal is to understand "why" another practitioner made a particular decision or choice (e.g., medical diagnosis, financial risk assessment). In this sense, they imagined treating models like colleagues and using explainability dialogues to facilitate natural interactions between models and people. Respondents viewed such natural language conversations as more intuitive for understanding model decisions than writing and debugging cumbersome code to generate explanations. Further, they imagined explainability dialogues giving more context to the explanations, such as assessments of accuracy, descriptions of how to interpret the explanations, and uncertainty, much like people do in everyday conversations [Paek and Horvitz, 2000]. Finally, they viewed conversations happening in a context-dependent manner, where they could easily follow up on previous queries for additional clarification or further lines of questioning. We summarize the key desiderata agreed on by the respondents below, in order of requirements most respondents agreed was important, where (N/26) indicates the number that agreed:

- (24/26) The dialogue should eliminate the need to learn and write the commands for generating explanations.
- (24/26) The system should describe the accuracy of the explanation in the dialogues.
- (23/26) The system should preserve context and enable follow-up questions.
- (21/26) The responses should be provided in real-time.
- (17/26) The dialogue system should decide which explanations to run. Users should not have to ask for a specific explainability algorithm.

These desiderata capture key elements in respondents' ultimate goals of engaging in conversations with machine learning models. For instance, respondents were excited about explainability dialogues involving natural, everyday questions to machine learning models such as, ”why did you make this decision?” and therefore agreed such systems should eliminate the need to write code, take the conversation context into account, and happen in real-time. Overall, respondents felt that natural language explainability dialogues would greatly improve their experiences using explanations and had clear ideas about how such a system should behave.

6.2 TalkToModel

The interviews demonstrated that natural language dialogues are a promising solution for supporting broad and accessible interactions with ML models due to their ease of use, capacity, and support for continuous discussion. However, designing a dialogue system that enables a satisfying model understanding experience introduces several challenges. First, the system must handle many conversation topics about the model and data while facilitating natural conversation flow [Ward and DeVault, 2017]. For instance, these topics may include explainability questions like the most important features for predictions and general questions such as data statistics or model errors. Further, the system must work for a variety of model classes and data, and it should understand language usage across different settings [Carenini et al., 1994]. For example, participants will use different terminology in conversations about loan prediction compared to disease diagnosis. Last, the dialogue system should generate accurate responses that address the users' core questions [Pennebaker et al., 2002, Zhang et al., 2020]. In the literature, researchers have suggested some prototype designs for generating explanations using natural language. However, these initial designs address specific explanations and model classes, limiting their applicability in general conversational explainability settings [Sokol and Flach, 2018, Feldhus et al., 2022b].

We address these challenges by introducing TalkToModel: a system that enables open-ended natural language dialogues for understanding ML models for any tabular dataset and classifier (an overview of TalkToModel is provided in Figure 7.1). Users can have discussions with TalkToModel about why predictions occur, how the predictions would change if the data changes, and how to flip predictions, among many other conversation topics (an example conversation is provided in Table 6.1). Further, they can perform these analyses on any group in the data, such as a single instance or a specific group of instances. For example, on a disease prediction task, users can ask "how important is BMI for the predictions?" or "so how would decreasing the glucose levels by ten change the likelihood of men older than twenty having the disease?" TalkToModel will respond by describing how, for instance, BMI is the most important feature for predictions, and decreasing glucose will decrease the chance by 20%. From there, users can engage further in the conversation by asking follow up questions like, "what if you instead increased glucose by ten for that group of men?" and TalkToModel use the context to accurately respond. Conversations with TalkToModel make

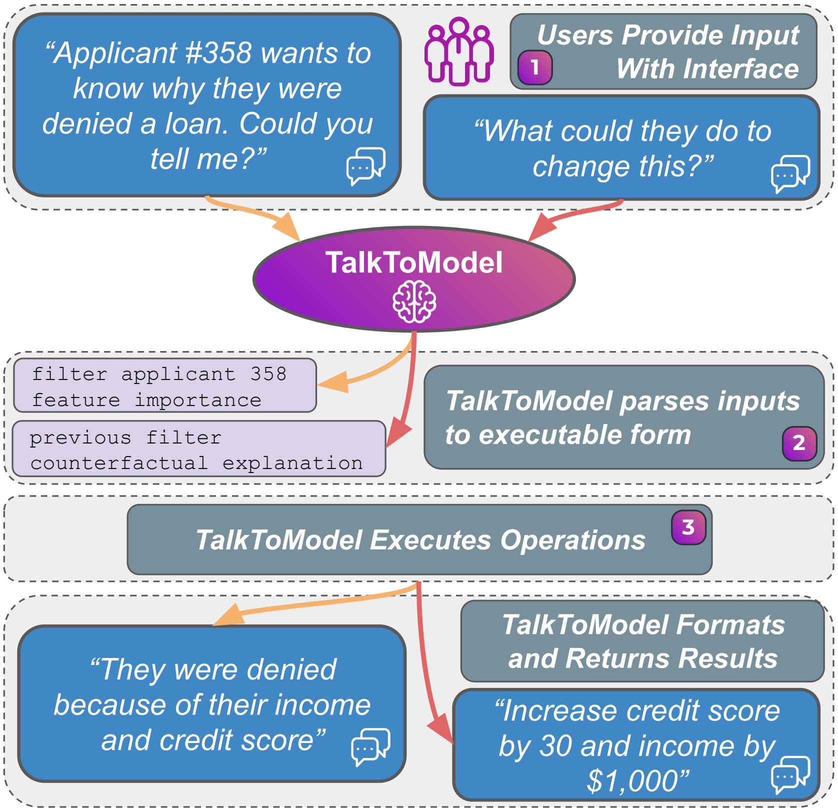

"Applicant #358 wants to know why they were denied a loan. Could you tell me?"
Users Provide Input With Interface
1
"What could they do to change this?"
TalkToModel
filter applicant 358 feature importance
previous filter counterfactual explanation
TalkToModel parses inputs to executable form
2
TalkToModel Executes Operations
3
"They were denied because of their income and credit score"
TalkToModel Formats and Returns Results
"Increase credit score by 30 and income by $1,000"

Figure 6.1: **Overview of TalkToModel:** Instead of writing code, users have conversations with TalkToModel as follows: **(1)** users supply natural language inputs. **(2)** the dialogue engine parses the input into an executable representation. **(3)** the execution engine runs the operations and the dialogue engine uses the results in its response.

model explainability straightforward because users can simply talk with the system in natural language about the model, and the system will generate useful responses.

To support such rich conversations with TalkToModel, we introduce techniques for both language understanding and model explainability. First, we propose a *dialogue engine* that parses user text inputs (referred to as *user utterances*) into an SQL-like programming language using a large language model (LLM). The LLM performs the parsing by treating the task of translating user utterances into the programming language as a seq2seq learning problem, where the user utterances are the source and parses in the programming language are the targets [Sutskever et al., 2014]. In addition, the TalkToModel language combines operations for explanations, ML error analyses, data manipulation, and descriptive text into a single language capable of representing a wide-variety of potential conversation topics for most model explainability needs (an overview of the different operations is provided in Table 6.2). To support the system adapting to any dataset and model, we introduce lightweight adaption techniques to fine-tune LLMs to perform the parsing, enabling strong generalization to new settings. Second, we introduce an *execution engine* that runs the operations in each parse. To reduce the burden of users deciding which explanations to run, we introduce methods that automatically select explanations for the user. In particular, this engine runs many explanations, compares their fidelities, and selects the most accurate ones. Finally, we construct a *text interface* where users can engage in open-ended dialogues using the system, enabling anyone, including those with minimal technical skills, to understand ML models.

6.3 Methods

In this section, we describe the components of TalkToModel. First, we introduce the dialogue engine and discuss how it understands user inputs, maps them to operations, and generates text responses based on the results of running the operations. Second, we describe the execution engine, which runs the operations. Finally, we provide an overview of the interface and the extensibility of TalkToModel.

6.3.1 Text Understanding

To understand the intent behind user utterances, the system learns to translate or *parse* them into logical forms. These parses represent the intentions behind user utterances in a highly-expressive and structured programming language TalkToModel executes.

Compared to dialogue systems that execute specific tasks by modifying representations of the internal state of the conversation [Chen et al., 2017, Li et al., 2017], our parsing-based approach allows for more flexibility in the conversations, supporting open-ended discovery, which is critical for model understanding. Also, this strategy produces a structured representation of user utterances instead of open-ended systems that generate unstructured free text [Santhanam and Shaikh, 2019]. Having this structured representation of user inputs is key for our setting where we need to execute specific operations depending on the user's input, which would not be straightforward with unstructured text.

TalkToModel performs the following steps to accomplish this: **1)** the system constructs a grammar for the user-provided dataset and model, which defines the set of acceptable parses, **2)** TalkToModel generates (utterance, parse) pairs for the dataset and model, **3)** the system finetunes a large language model (LLM) to translate user utterances into parses, and **4)** the system responds conversationally to users by composing the results of the executed parse into a response that provides context for the results and opportunities to follow up.

Grammar To represent the intentions behind the user utterances in a structured form, TalkToModel relies on a grammar, defining a domain specific language for model understanding. While the user utterances themselves will be highly diverse, the grammar creates a way to express user utterances in a structured yet highly expressive fashion that the system can reliably execute. Compared with approaches that treat determining user intentions in conversations as a classification problem [Liu et al., 2017, Cai and Chen, 2020], using a grammar enables the system to express compositions of operations and arguments that take on many different values, such as real numbers, that would otherwise be combinatorially impossible in a prediction setting. Instead, TalkToModel translates user utterances into this grammar in a seq2seq fashion, overcoming these challenges [Sutskever et al., 2014]. This grammar consists of production rules that include the `operations` the system can run (an overview is provided in Table 6.2), the acceptable `arguments` for each operation, and the relations between operations. One complication is that user-provided datasets have different feature names and values, making it hard to define one shared grammar between datasets. Instead, we update the grammar based on the feature names and values in a new dataset. For instance, if a dataset only contained the feature names `age` and `income`, these two names would be the only acceptable values for the `feature` argument in the grammar.

To ensure our grammar provides sufficient coverage for XAI questions, we evaluate whether our grammar supports the questions from the *XAI question bank*. This question bank was introduced by Liao et al. [2020] based on interviews with AI product designers and includes 31 core, prototypical questions XAI systems should answer, excluding socio-technical questions beyond the scope of TalkToModel (e.g., What are the results of other people using the [model]). The prototypical questions address topics such as the input/data to the model ("What is the distribution of a given feature?"), model output ("What kind of output does the system give?"), model performance ("How accurate are the predictions?"), global model behavior ("What is the system's overall logic?"), why/why not the system makes individual predictions ("Why is this instance predicted this class?"), and what-if or counterfactual questions ("What would the system predict if this instance changes to...?"). To evaluate how well TalkToModel covers these questions, we review each question and evaluate whether our grammar can parse it. Overall, we find our grammar supports 30/31 of the prototypical questions. We provide a table of each question and corresponding parse in Appendix Table B.1 and Supplementary Table B.2. Overall, the grammar covers the vast majority of XAI related questions, and therefore, has good coverage of XAI topics.

Supporting Context In Dialogues User conversations with TalkToModel naturally include complex conversational phenomena such as anaphora and ellipsis [Grosz et al., 1983, Tseng et al., 2021, Guo et al., 2018]. Meaning, conversations refer back to events earlier in the conversation ("what do you predict for *them*?") or omit information that must be inferred from conversation ("Now show me for people predicted incorrectly."). However, current

Table 6.1: A conversation about diabetes prediction, demonstrating the breadth of different conversation points the system can discuss.

User “utterances” & TalkToModel response
“What are the predictions on patients older than 30? Could you also tell me what would happen to these predictions if we increased bmi by 10?”
On data with **age greater than 30.0**, the model predicts: ➩ **No Diabetes** on 46.8% of data ➩ **Diabetes** on 53.2% of data On data with **age greater than 30.0 and bmi is increased by 10.0,** the model predicts: ➩ **No Diabetes** on 30.1% of data ➩ **Diabetes** on 69.9% of data
“So how important is bmi for the model’s predictions on the over thirty group?”
For the model’s predictions on data with **age greater than 30.0**, the **bmi** feature is ranked on average **2.86** (95% CI [2.685, 3.035]). Here, rank 1 is the most important feature (out of 8 features). Overall, **bmi** is **highly important** for the model’s predictions.
“Ah, so which two features are mostly more important than bmi for these people?”
For the models predictions on data **where age is greater than 30.0**, the **top 2** most important features are: 1. **glucose** 2. **pedigree_function**
“What types of patients is the model typically predicting incorrect here”?
For data with **age greater than 30.0**, the model typically predict predicts incorrect: ➩ if (**bmi** > **26.95**) and (**glucose** ≤ **125.0**) and (**pedigree_function** then the model is incorrect on *62.3%* of *53* samples ➩ (**bmi** > **e26.95**) and (**glucose** > **125.0**) and (**age** > **57.5**) **¡= 0.943**) then the model is incorrect on *62.5%* of *8* samples ➩ (**bmi** ≤ **26.95**) and (**glucose** > **180.5**) then the model is incorrect on *100.0%* of *2* samples. Want to take a closer look at these rules?

language models only parse a single input, making it hard to apply them in settings where the context is important. To support context in the dialogues, TalkToModel introduces a set of operations in the grammar that determine the context for user utterances. In contrast with approaches that maintain the conversation state using neural representations [Chen et al., 2017, Gao et al., 2019], grammar operations allow for much more trustworthy and dependable behavior while still fostering rich interactions, which is critical for high-stakes settings, and similar mechanisms for incorporating grammar predicates across turns have been shown to achieve strong results [Guo et al., 2018]. In particular, we leverage two operations: `previous filter` and `previous operation`, which look back in the conversation to find the last filter and last operation, respectively. These operations also act recursively. Therefore, if the last filter is a `previous filter` operation, TalkToModel will recursively call `previous filter` to resolve the entire stack of filters. As a result, TalkToModel is capable of addressing instances of anaphora and ellipsis by using these operations to resolve the entity via co-reference or infer it from the previous conversation history. This dynamic enables users to have complex and natural conversations with TalkToModel.

Parsing Dataset Generation To parse user utterances into the grammar, we finetune an LLM to translate utterances into the grammar in a seq2seq fashion. We use LLMs because these models have been trained on large amounts of text data and are robust priors for language understanding tasks. Thus, they can better understand diverse user inputs than training from scratch, improving the user experience. Further, we automate the finetuning of an LLM to parse user utterances into the grammar by generating a training dataset of (utterance, parse) pairs. Compared to dataset generation methods that use human annotators to generate and label datasets for training conversation models [Gao et al., 2018, Rieser and Lemon, 2012], this approach is much less costly and time consuming, while still being highly effective, and supports users getting conversations running very quickly. This strategy consists of writing an initial set of user utterances and parses, where parts of the utterances and parses are *wildcard* terms. TalkToModel enumerates the wildcards with aspects of a user-provided dataset, such as the feature names, to generate a training dataset. Depending on the user-provided dataset schema, TalkToModel typically generates anywhere from $20,000$-

40, 000 pairs. Last, we have already written the initial set of utterances and parses, so users only need to provide their dataset to setup a conversation.

Semantic Parsing Here, we provide additional details about the semantic parsing approach for translating user utterances into the grammar. The two strategies for parsing user utterances using pre-trained LLMs that we considered were **1.)** few-shot GPT-J [Wang and Komatsuzaki, 2021] and **2.)** finetuned T5 [Raffel et al., 2020]. With respect to the few-shot models, because the LLM's context window only accepts a fixed number of inputs, we introduce a technique to select the set of most relevant prompts for the user utterance. In particular, we embed all the utterances and identify the closest utterances to the user utterance according to the cosine distance of these embeddings. To ensure a diverse set of prompts, we only select one prompt per template. We prompt the LLM using these (utterance, parse) pairs, ordering the closest pairs immediately before the user utterance because LLMs exhibit recency biases [Zhao et al., 2021]. Using this strategy, we experiment with the number of prompts included in the LLM's context window. In practice, we use the `all-mpnet-base-v2` sentence transformer model to perform the embeddings [Reimers and Gurevych, 2019], and we consider the GPT-J 6B, GPT-Neo 2.7B, and GPT-Neo 1.3B models in our experiments.

We also fine-tune pre-trained T5 models in a seq2seq fashion on our datasets. To perform fine-tuning, we split the dataset using a 90/10% train/validation split and train for 20 epochs to maximize the next token likelihood with a batch size of 32. We select the model with the lowest validation loss at the end of each epoch. We fine-tune with a learning rate of $1e$-4 and the AdamW optimizer [Loshchilov and Hutter, 2019]. Last, our experiments consider the T5 Small, Base, and Large variants.

Generating Responses After TalkToModel executes a parse, it composes the results of the operations into a natural language response it returns to the user. TalkToModel generates these responses by filling in templates associated with each operation based on the results. The responses also include sufficient context to understand the results and opportunities for following up (examples in Table 6.1). Further, because the system runs multiple operations in one execution, TalkToModel joins response templates, ensuring semantic coherence, into a final response and shows it to the user. Compared to approaches that generate responses using neural methods [Shao et al., 2017], this approach ensures the responses are trustworthy and do not contain useless information hallucinated by the system, which would be a very poor user experience for the high-stakes applications we consider. Further, because TalkToModel supports a wide variety of different operations, this approach ensures sufficient diversity in responses, so they are not repetitive.

6.3.2 Executing Parses

In this subsection, we provide an overview of the execution engine, which runs the operations necessary to respond to user utterances in the conversation. Further, this component automatically selects the most faithful explanations for the user, helping ensure explanation accuracy.

Feature Importance Explanations At its core, TalkToModel explains why the model makes predictions to users with feature importance explanations. We implement the feature importance explanations using *post-hoc* feature importance explanations. Post-hoc feature importance explanations do not rely on internal details of the model f (e.g., internal weights or gradients) and only on the input data $\mathbf{x}$ and predictions y to compute explanations, so users are not limited to only certain types of models [Ribeiro et al., 2016b, Lundberg et al., 2020, Lakkaraju et al., 2019, Plumb et al., 2018, Li et al., 2021]. Note that our system can easily be extended to other explanations that rely on internal model details, if required [Selvaraju et al., 2017, Murdoch et al., 2019, Agarwal et al., 2021a, Sundararajan et al., 2017].

Table 6.2: Overview of the *operations* supported by TalkToModel, which are incorporated into the conversation to generate responses.

	`operation`, `arguments`, **and description**
Data	`filter(dataset, feature, value, comparison)`: filters `dataset` by using value and comparison operator
	`change(dataset, feature, value, variation)`: Changes `dataset` by increasing, decreasing, or setting feature by `value`
	`show(list)`: Shows items in list in the conversation
	`statistic(dataset, metric, feature)`: Computes summary statistic for `feature`
	`count(list)`: Length of list
	`and(op1, op2)`: Logical "and" of two operations
	`or(op1, op2)`: Logical "or" of two operations
Explainability	`explain(dataset, method, class=predicted)`: Feature importances on `dataset`
	`cfe(dataset, number, class=opposite)`: Gets `number` counterfactual explanations
	`topk(dataset, k)`: Top `k` most important features
	`important(dataset, feature)`: Importance ranking of `feature`
	`interaction(dataset)`: Interaction effects between features
	`mistakes(dataset)`: Patterns in the model's errors on `dataset`
ML	`predict(dataset)`: Model predictions on `dataset`
	`likelihood(dataset)`: Prediction probabilities on `dataset`
	`incorrect(dataset)`: Incorrect predictions
	`score(dataset, metric)`: Scores the model with `metric`
Conv.	`prev_filter(conversation)`: Gets last filters
	`prev_operation(conversation)`: Gets last non-filtering operations
	`followup(conversation)`: Respond to system followups
Description	`function()`: Overview of the system's capabilities
	`data(dataset)`: Summary of dataset
	`model()`: Description of `model`
	`define(term)`: Defines `term`

Explanation Selection While there exists several post hoc explanation methods, each one adopts a different definition of what constitutes an explanation [Krishna et al., 2022]. For example, while LIME, SHAP, and Integrated Gradients all output feature attributions, LIME returns coefficients of local linear models, SHAP computes Shapley values, and Integrated Gradients leverages model gradients. Consequently, we automatically select the *most faithful* explanation for users, unless a user specifically requests a certain technique. Following prior works, we compute faithfulness by perturbing the most important features and evaluating how much the prediction changes [Meng et al., 2022]. Intuitively, if the feature importance ϕ correctly captures the feature importance ranking, perturbing more important features should lead to greater effects.

While previous works [Lundberg et al., 2020, Hooker et al., 2019], compute the faithfulness over many different thresholds, making comparisons harder, or require retraining entirely from scratch, we introduce a single metric that captures the prediction sensitivity to perturbing certain features called the *fudge score*. This metric is the mean absolute difference between the model's prediction on the original input and a fudged version on $\mathbf{m} \in \{0,1\}^d$ features,

$$\text{Fudge}(f, \mathbf{x}, \mathbf{m}) = \frac{1}{N} \sum_{n=1}^{N} |f(\mathbf{x}) - f(\mathbf{x} + \epsilon_n \odot \mathbf{m})| \tag{6.1}$$

where $\odot$ is the tensor product and $\epsilon \sim \mathcal{N}(0, I\sigma)$ is $N \times d$ dimensional Gaussian noise. To evaluate faithfulness for a particular explanation method, we compute area under the fudge score curve on the top-k most important features, thereby summarizing the results into a single metric,

$$\mathbb{1}(k, \phi) = \begin{cases} 1 & \text{if } \phi_i \in \arg\max_{\phi' \subset \{1..d\}, |\phi'| = k} \sum_{i \in \phi'} |\phi_i| \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

$$\text{Faith}(\phi,\ f,\ \mathbf{x},\ K) = \sum_{k=1}^{K} \text{Fudge}(f,\ \mathbf{x},\ \mathbb{1}(k, \phi)) \tag{6.3}$$

where $\mathbb{1}(k, \phi)$ is the indicator function for the top-k most important features. Intuitively, if a set of feature importances ϕ correctly identifies the most important features, perturbing them will have greater effects on the model's predictions, resulting in higher faithfulness scores. We compute faithfulness for multiple different explanations and select the highest. In practice, we consider LIME [Ribeiro et al., 2016b] with the following kernel widths $[0.25, 0.50, 0.75, 1.0]$

and KernelSHAP [Lundberg and Lee, 2017b]. We leave all settings to default besides the kernel widths for LIME. In practice, we set $\sigma = 0.05$ to ensure perturbations happen in the local region around the prediction, K to floor($\frac{d}{2}$), and $N = 10,000$ to sample sufficiently. One complication arises for categorical features, where we cannot apply Gaussian perturbations. For these features, we randomly sample these features from a value in the dataset column 30% of the time to guarantee the feature remains categorical under perturbation. Last, if multiple explanations return similar fidelities, we use the explanation stability metric proposed by Alvarez-Melis and Jaakkola [2018] to break ties, because it is much more desirable for the explanation to robust to perturbations [Slack et al., 2021a, Agarwal et al., 2022a]. In order to use the *stability* metric proposed by Alvarez-Melis and Jaakkola [2018] to break ties if the explanations fidelities are quite close (less than $\delta = 0.01$), we compute the jaccard similarity between feature rankings instead of the $l2$ norm as is used in their work. The reason is that the norm might not be comparable between explanation types, because they have different ranges, while the jaccard similarity should not be affected. Further, we compute the area under the top k curve using the jaccard similarity stability metric, as in Equation 6.3, to make this measure more robust.

Additional Explanation Types Since users will have explainability questions that cannot be answered solely with feature importance explanations, we include additional explanations to support a wider array of conversation topics. In particular, we support counterfactual explanations and feature interaction effects. These methods enable conversations about *how* to get different outcomes and if features *interact* with each other during predictions, supporting a broad set of user queries. We implement counterfactual explanations using DiCE, which generates a diverse set of counterfactuals [Mothilal et al., 2020]. Having access to many plausible counterfactuals is desirable because it enables users to see a breadth of different, potentially useful, options. Also, we implement feature interaction effects using the partial dependence based approach from Greenwell et al. [2018] because it is effective and quick to compute.

Exploring Data and Predictions Because the process of understanding models often requires users to inspect the model's predictions, errors, and the data itself, TalkToModel supports a wide variety of data and model exploration tools. For example, TalkToModel provides options for filtering data and performing what-if analyses, supporting user queries that concern subsets of data or what would happen if data points change. Users can also inspect model errors, predictions, prediction probabilities, compute summary statistics, and evaluation metrics for individuals and groups of instances. TalkToModel additionally supports summarizing common patterns in mistakes on groups of instances by training a shallow decision tree on the model errors in the group. Also, TalkToModel enables descriptive operations, which explain how the system works, summarize the dataset, and define terms to help users understand how to approach the conversation. Overall, TalkToModel supports a rich set of conversation topics in addition to explanations, making the system a complete solution for the model understanding requirements of end users.

Extensibility

While we implement TalkToModel with several different choices for operations such as feature importance explanations and counterfactual explanations, TalkToModel is highly modular and system designers can easily incorporate new operations or change existing ones by modifying the grammar to best support their user populations. This design makes TalkToModel straightforward to extend to new settings, where different operations may be desired.

Table 6.3: Exact Match Parsing Accuracy (%) for the 3 gold datasets, on the IID and Compositional splits, as well as Overall. The fine-tuned T5 models perform significantly better than few-shot GPT-J, and T5 Large performed the best. These results demonstrate that TalkToModel can understand user intentions with a high degree of accuracy using the T5 models.

	German			Compas			Diabetes		
	IID	Comp.	Overall	IID	Comp.	Overall	IID	Comp.	Overall
Nearest Neighbors	26.2	0.0	16.5	27.4	0.0	21.9	10.9	0.0	8.4
GPT-Neo 1.3B									
10-shot	41.3	4.1	27.5	35.9	0.0	28.8	40.1	7.0	32.6
20-shot	39.7	0.0	25.0	39.3	0.0	31.5	42.9	2.3	33.7
30-shot	42.9	0.0	27.0	39.3	0.0	31.5	41.5	4.7	33.2
GPT-Neo 2.7B									
5-shot	38.1	4.1	25.5	35.9	3.4	29.5	46.9	7.0	37.9
10-shot	38.1	6.8	26.5	40.2	3.4	32.9	40.8	9.3	33.7
20-shot	39.7	0.0	25.0	39.3	0.0	31.5	42.9	2.3	33.7
GPT-J 6B									
5-shot	51.6	14.9	38.0	51.3	6.9	42.5	55.8	7.0	44.7
10-shot	57.9	9.5	40.0	49.6	3.4	40.4	53.7	9.3	43.7
T5									
small	61.1	32.4	50.5	71.8	10.3	59.6	77.6	30.2	66.8
base	68.3	**48.6**	61.0	65.0	10.3	54.1	**84.4**	34.9	73.2
large	**74.6**	44.6	**63.5**	**76.9**	**24.1**	**66.4**	**84.4**	**51.2**	**76.8**

6.4 Results

In this section, we demonstrate TalkToModel accurately understands users in conversations by evaluating its language understanding capabilities on ground truth data. Next, we evaluate the effectiveness of TalkToModel for model understanding by performing a real-world human study on healthcare workers (e.g., doctors and nurses) and ML practitioners, where we benchmark TalkToModel against existing explainability systems. We find users both prefer and are more effective using TalkToModel than traditional point-and-click explainability systems, demonstrating its effectiveness for understanding ML models.

Language Understanding

Here, we quantitatively assess the language understanding capabilities of TalkToModel by creating gold parse datasets and evaluating the system's accuracy on this data.

Gold Parse Collection We construct gold datasets (i.e., ground truth (utterance, parse) pairs) across multiple datasets to evaluate the language understanding performance of our models. To construct these gold datasets, we adopt an approach inspired by Yu et al. [2018], which constructs a similar dataset for multitask semantic parsing.

Our gold dataset generation process is as follows. First, we write 50 (utterance, parse) pairs for the particular task (i.e., loan or diabetes prediction). These utterances range from simple "How likely are people in the data to have diabetes?" to complex "If these people were not unemployed, what's the likelihood they are good credit risk? Why?" and conversational "What if they were twenty years older?". We include each operation (Table 6.2) at least twice in the parses, to make sure there is good coverage. From there, we ask Mechanical Turk workers to rewrite the utterances while preserving their semantic meaning to ensure that the ground truth parse for the revised utterance is the same but the phrasing differs. We ask workers to rewrite each pair 8 times for a total of 400 (utterance, parse) pairs per task. Next, we filter out low-quality mturk revisions. We ask the crowd sourced workers to rate the similarity between the original utterance and revised utterance on a scale of (1-4), where 4 indicates the utterances have the same meaning and 1 that they do not have the same meaning. We collect 5 ratings per revision and remove (utterance, parse) pairs that score below 3.0 on average. Finally, we perform an additional filtering step to ensure data quality by inspecting the remaining pairs ourselves and removing any bad revisions.

Since we want to evaluate TalkToModel's capacity to generalize across different scenarios, we perform this data collection process across 3 different tasks: Pima Indian Diabetes

Dataset [Dua and Graff, 2017], German Credit Dataset [Dua and Graff, 2017], and the COMPAS recidivism dataset [Angwin et al., 2016]. After collecting revisions and ensuring quality, we are left with 200 pairs for German Credit, 190 for diabetes, and 146 for COMPAS.

Models We compare two strategies for using pre-trained LLMs to parse user utterances into the grammar **1.)** few-shot GPT-J [Wang and Komatsuzaki, 2021] and **2.)** finetuned T5 [Raffel et al., 2020]. Both these models translate user utterances into the TalkToModel grammar in a seq2seq fashion. However, the GPT-J models are higher-capacity and more amenable to be trained by in-context learning. This procedure includes examples of the input and target from the training prepended to the test instance [Brown et al., 2020, Min et al., 2022, Xie et al., 2022]. On the other hand, the T5 models require traditional finetuning on the input and target pairs. Consequently, the few-shot approach is quicker to set up because it does not require finetuning, making it easier for users to get started with the system. However, the finetuned T5 leads to improved performance and a better user experience overall while taking longer to set up. To train these models through finetuning or prompting, we generate synthetic (utterance, parse) pairs because it is impractical to assume that we can collect ground truth pairs for every new task we wish to use TalkToModel. We provide additional training details in the methods section.

We evaluate both fine-tuned T5 models and few-shot GPT-J models on the testing data. We additionally implement a naive nearest neighbors baseline, where we select the closest user utterance in the synthetic training set according to cosine distance of `all-mpnet-base-v2` sentence embeddings and return the corresponding parse [Reimers and Gurevych, 2019]. For the GPT-J models, we compare N-Shot performance, where N is the number of (utterance, parse) pairs from the synthetically generated training sets included in the prompt, and sweep over a range of N for each model. For the larger models, we have to use relatively smaller N in order for inference to fit on a single 48GB GPU.

When parsing the utterances, one issue is that their generations are unconstrained and may generate parses outside the grammar, resulting in the system failing to run the parse and bad user experiences. To ensure the generations are grammatical, we constrain the decodings to be in the grammar [Shin et al., 2021]. This technique, referred to as *guided decoding*, constrains the LLM generations to only allow those tokens that appear next in the grammar at any point during generation. Practically, we accomplish this by recompiling the grammar at inference time into an equivalent grammar consisting of the tokens in the LLM's vocabulary. While decoding from the LLM, we fix the likelihood of ungrammatical tokens to 0 at every generation step. Thus, the LLM only generates grammatical parses.

Evaluating The Parsing Accuracy To evaluate performance on the datasets, we use the exact match parsing accuracy [Talmor et al., 2017, Yu et al., 2018, Gupta et al., 2022]. This metric is whether the parse exactly matches the gold parse in the dataset. In addition, we perform the evaluation on two splits of each gold parse dataset, in addition to the overall dataset. These splits are the IID and compositional splits. The IID split contains (utterance, parse) pairs where the parse's `operations` and their structure (but not necessarily the arguments) are in the training data. The compositional split consists of the remaining parses that are not in the training data. Because LM's struggle compositionally, this split is generally much harder for LM's to parse [Oren et al., 2020, Yin et al., 2021].

Accuracy We present the results in Table 6.3. The fine-tuned T5 performs better overall than the few shot GPT-J models. In particular, the T5 Large models perform strongly on both the IID and compositional data and can even parse complex compositional phrases. Notably, the T5 small model performs better than the GPT-J 6B model, which has two orders of magnitude more parameters. This dynamic is particularly true in the compositional splits in the data where the GPT-J few shot models never exceed 10% parsing accuracy. Overall, these results indicate TalkToModel can understand user utterances with a high degree of accuracy using our best performing T5 models. Further, we recommend using this model for the best results and use it for our remaining evaluation.

6.4.1 Advantages of Explanation Selection

We introduce a technique for explanation selection, which we used in TalkToModel to automatically select high-quality explanations for the conversation (Subsection 6.3.2). In this appendix, we provide more details about the advantages of explanation selection. In particular, we rigorously benchmark our selection method against SOTA explanation techniques like LIME and SHAP. We show that our explanation selection method computes the most faithful explanations.

To perform this analysis, we use the faithfulness metrics provided by the widely-used OpenXAI framework [Agarwal et al., 2022b]. Specifically, we use the Prediction Gap on Important feature perturbation (PGI) and the Prediction Gap on Unimportant feature perturbation (PGU) metrics. These metrics measure the change in perturbing the most influential features and least important features, respectively. Intuitively, PGI captures that perturbations to influential features should result in more significant changes to predictions (higher PGI is better). PGU captures that perturbations to non-influential features should result in smaller changes to the prediction (lower PGU is better).

We compare our explanation selection method against both SOTA explanation methods LIME and SHAP [Ribeiro et al., 2016b, Lundberg and Lee, 2017b]. To make our evaluation more comprehensive, we compare against LIME using 4 different settings of the kernel width hyperparameter $[0.25, 0.50, 0.75, 1.0]$, because this hyperparameter can have significant effects on the resulting explanation. We leave all settings to default otherwise. Further, we perform this comparison using our 3 diverse datasets: Diabetes, COMPAS, and German Credit, and we compute explanations three times for each data point to reduce error due to explanation

Table 6.4: The prediction gap on important features (PGI) and prediction gap on unimportant features (PGU) results. We bold the statistically significant best result. Overall, explanation selection is the best explanation method in all settings, except for PGU and the german credit data where it is better than SHAP but not significantly better than LIME.

	PGI↑			PGU↓		
	Diabetes	COMPAS	German	Diabetes	COMPAS	German
LIME, Width=0.25	0.070	0.124	3.897	0.032	0.031	**0.774**
LIME, Width=0.50	0.072	0.127	3.871	0.020	0.027	**0.793**
LIME, Width=0.75	0.071	0.127	3.856	0.021	0.026	**0.808**
LIME, Width=1.0	0.070	0.127	3.853	0.022	0.026	**0.799**
SHAP	0.083	0.117	2.094	0.031	0.031	3.007
Explanation Selection	**0.107**	**0.138**	**4.011**	**0.006**	**0.023**	**0.788**

sampling. We set the important features used for the PGI metric to the most influential 50% of features and the unimportant features used for the PGU metric to the least influential 50% of features to ensure we provide a comprehensive evaluation for the explanation's ranking of all features in the data.

We present the results in Table 6.4 and provide the mean PGU or PGI value for each explanation and dataset. Further, we bold the best statistically significant result according to a Bonferroni corrected t-test, where we compare the explanation selection procedure to each of the other explanation methods for the respective dataset and metric. Overall, we find that explanation selection performs better than baseline SOTA explanations across almost every dataset and metric considered, except for the PGU metric on the German dataset, where explanation selection performs on par with the best performing LIME explanations.

User Study: Utility of Explainability Dialogues

The results in the previous subsection show TalkToModel understands user intentions to a high degree of accuracy. In this subsection, we evaluate how well the end-to-end system helps users understand ML models compared to current explainability systems.

Study Overview We compare TalkToModel against *explainerdashboard*, one of the most popular open-source explainability dashboards [Dijk et al., 2022]. This dashboard has similar functionality to TalkToModel, considering it provides an accessible way to compute explanations and perform model analyses. Thus, it is a reasonable baseline. Last, we perform this comparison using the Diabetes dataset, and a gradient boosted tree trained on the data [Pedregosa et al., 2011]. To compare both systems in a controlled manner, we ask participants to answer general ML questions with TalkToModel and the dashboard. Each question is about basic explainability and model analysis, and participants answer using multiple choice, where one of the options is "Could not determine." if they cannot figure out the answer (though it is straightforward to answer all the questions with both interfaces). For example, questions are about comparing feature importances "Is glucose more important than age for the model's predictions for data point 49?" or model predictions "How many people are predicted not to have diabetes but do not actually have it?" Participants answer 10 total questions. We divide the 10 questions into 2 blocks of 5 questions each. Both blocks have similar questions but different values to control for memorization. Participants use TalkToModel to answer one block of questions and the dashboard for the other block. In addition, we provide a tutorial on how to use both systems before showing users the questions for the system. Last, we randomize question, block, and interface order to control for biases due to showing interfaces or questions first.

Metrics Following previous work on evaluating human and ML coordination and trust, we assess several metrics to evaluate user experiences [Chen et al., 2022, Freed et al., 2008, Glass et al., 2008]. We evaluate the following statements along 1-7 Likert scale at the end of the survey:

- **Easiness:** *I found the conversational interface easier to use than the dashboard interface*
- **Confidence:** *I was more confident in my answers using the conversational interface*

Table 6.5: User study results: % of respondents that agree (> Neutral Likert score) TalkToModel is better than the dashboard in the 4 comparison questions. A significant portion of respondents agreed TalkToModel is better than the dashboard in all the categories except Grad. students and "Likeliness To Use". Still, a majority agreed TalkToModel was superior in this case.

	% Agree TalkToModel Better	
Comparison	Health Care Workers	ML Grad. Students
Easiness	82.2	84.6
Confidence	77.7	69.2
Speed	84.4	84.6
Likeliness To Use	73.3	53.8

than the dashboard interface

- **Speed:** *I felt that I was able to more rapidly arrive at an answer using the conversational interface than the dashboard interface*
- **Likeliness To Use:** *Based on my experience so far with both interfaces, I would be more likely to use the conversational interface than the dashboard interface in the future*

To control for bias associated with the ordering of the terms conversational interface and dashboard interface, we randomized their ordering. We also measure accuracy and time-taken to answer each question. Last, we asked to participants to write a short description comparing their experience with both interfaces to capture participants qualitative feedback about both systems.

Recruitment Since TalkToModel provides an accessible way to understand ML models, we expect it to be useful for subject matter experts with a variety of experience in ML, including users without any ML experience. As such, we recruited 45 English speaking healthcare workers to take the survey using the Prolific service [Palan and Schitter, 2018] with minimal or no ML expertise This group comprises a range of healthcare workers, including doctors, pharmacists, dentists, psychiatrists, healthcare project managers, and medical scribes. The vast majority of this group (43) stated they had either no experience with ML or had heard about it from reading articles online, while two members indicated they had equivalent to an undergraduate course in ML. As another point of comparison, we recruited ML professionals with relatively higher ML expertise from ML Slack channels and email lists. We received 13 potential participants, all of which had graduate course level ML experience or higher, and included all of them in the study. We received IRB approval for this study from our institution's IRB approval process and informed consent from participants.

Metric Results A significant majority of health care workers agreed they preferred TalkToModel in all the categories we evaluated (Table 6.5). The same is true for the ML professionals, save for whether they were more likely to use TalkToModel in the future, where 53.8% of participants agreed they would instead use TalkToModel in the future. In addition, participants subjective notions around how quickly they could use TalkToModel aligned with their actual speed of use, and both groups arrived at answers using TalkToModel significantly quicker than the dashboard. The median question answer time (measured at the total time taken from seeing the question to submitting the answer) using TalkToModel was 76.3 seconds, while it was 158.8 seconds using the dashboard.

Participants were also much more accurate and completed questions at a higher rate (i.e., they did not mark "could not determine) using TalkToModel (Table 6.6). While both health care workers and ML practitioners clicked could not determine for a quarter of the questions using the dashboard, this was true for 13.8% of health care workers and 6.1% of ML professionals using TalkToModel, demonstrating the usefulness of the conversational interface. On completed questions, both groups were much more accurate using TalkToModel than the dashboard. Most surprisingly, though ML professionals agreed they preferred TalkToModel only about half the time, they answered all the questions correctly using it, while they only answered 62.5% of questions correctly with the dashboard. Finally, we

Table 6.6: User study results: Completion rate and accuracy across interfaces and participant groups. We compute the completion rate as the questions users provided and answer for and did not mark "could not determine." We measure accuracy on completed questions. Participants answered questions at a higher rate more accurately using TalkToModel than the dashboard.

	% Questions Completed		% Accuracy On Completed Questions	
	Dash.	TalkToModel	Dash.	TalkToModel
Health Care Workers	74.7	86.2	66.1	91.8
ML Grad. Students	73.8	93.9	62.5	100.0

observed TalkToModel's conversational capabilities were highly effective. There were only 6 utterances out of over $1,000$ total utterances the conversational aspect of the system failed to resolve. These failure cases generally involved certain discourse aspects like asking for additional elaboration ("more description").

The largest source of errors for participants using the explainability dashboard were two questions concerning the top most important features for individual predictions. The errors for these questions account for 47.4% of health care workers and 44.4% of ML professionals' total mistakes. Answering these questions with the dashboard requires users to perform multiple steps, including choosing the feature importance tab in the dashboard, selecting local explanations for the correct instance, and ranking the features according to their importance. On the other hand, the streamlined text interface of TalkToModel made it much simpler to solve these questions resulting fewer errors.

Qualitative Results For the qualitative user feedback, we provide representative quotes from similar themes in the responses. Users expressed that they could more rapidly and easily arrive at results, which could be helpful for their professions,

> *I prefer the conversational interface because it helps arrive at the answer very*

> *quickly. This is very useful especially in the hospital setting where you have hundreds of patients getting check ups and screenings for diabetes because it is efficient and you can work with medical students on using the system to help patient outcomes.*—P39 medical worker at a tertiary hospital.

Participants also commented on the user friendliness of TalkToModel and its strong conversational capabilities, stating, "the conversational [interface] was straight to the point, way easier to use"—P35 Nurse, and that "the conversational interface is hands-down much easier to use... it feels like one is talking to a human."—P45 ML Professional. We did not find any negative feedback surrounding the conversational capabilities of the system. Users also commented on how easy it was to access information compared to the dashboard,

> *With the conversational interface you can ask whatever you want to know and with the dashboard you need to specifically search information that you don't actually know where it is.*—P31 Physical Therapist.

All in all, users expressed strong positive sentiment about TalkToModel due to the quality of conversations, presentation of information, accessibility, and speed of use.

Several ML professionals brought up points that could serve as future research directions. Notably, participants stated they would rather look at the data themselves rather than rely on an interface that rapidly provides an answer,

> *I would almost always rather look at the data myself and come to a conclusion than getting an answer within seconds.*—P11 ML Professional.

6.5 Summary of Contribution

In this chapter, we proposed the use of natural language conversations for explaining ML models and introduced an effective system to this end: TalkToModel. We showed TalkToModel makes explainable AI accessible to users that come from a range of backgrounds by using natural language conversations. Our experimental findings demonstrated TalkToModel both comprehends users to a high-degree of accuracy and can help users understand the predictions of ML models much better than existing systems. In particular, we showed TalkToModel is a highly effective way for domain experts such as healthcare workers to understand ML models, like those applied to disease diagnosis. Last, we designed TalkToModel to be highly extensible, making it straightforward for explainability users and researchers to build on the system. This chapter is based on the following publications:

- *Rethinking Explainability as a Dialogue: A Practitioner's Perspective* (Lakkaraju et al. [2022], HCAI @ NeurIPS 2022)
- *TalkToModel: Explaining Machine Learning Models with Interactive Natural Language Conversations* (Slack et al. [2022], TSRML @ NeurIPS 2022, honorable mention best paper award)

which have a combined 23 citations at the time of writing.

Himabindu Lakkaraju and the author of this dissertation share primary authorship of *Rethinking Explainability as a Dialogue: A Practitioner's Perspective.* Himabindu Lakkaraju was responsible for designing and performing the interviews, while the dissertation author performed the qualitative and quantitative evaluation of the results. This work also included extensive design and analysis of plausible explainabilty dialogue systems, which the dissertation author is responsible for, but these are omitted from the dissertation for continuity with the rest of the chapter. The author of this dissertation is the primary author of *TalkToModel:*

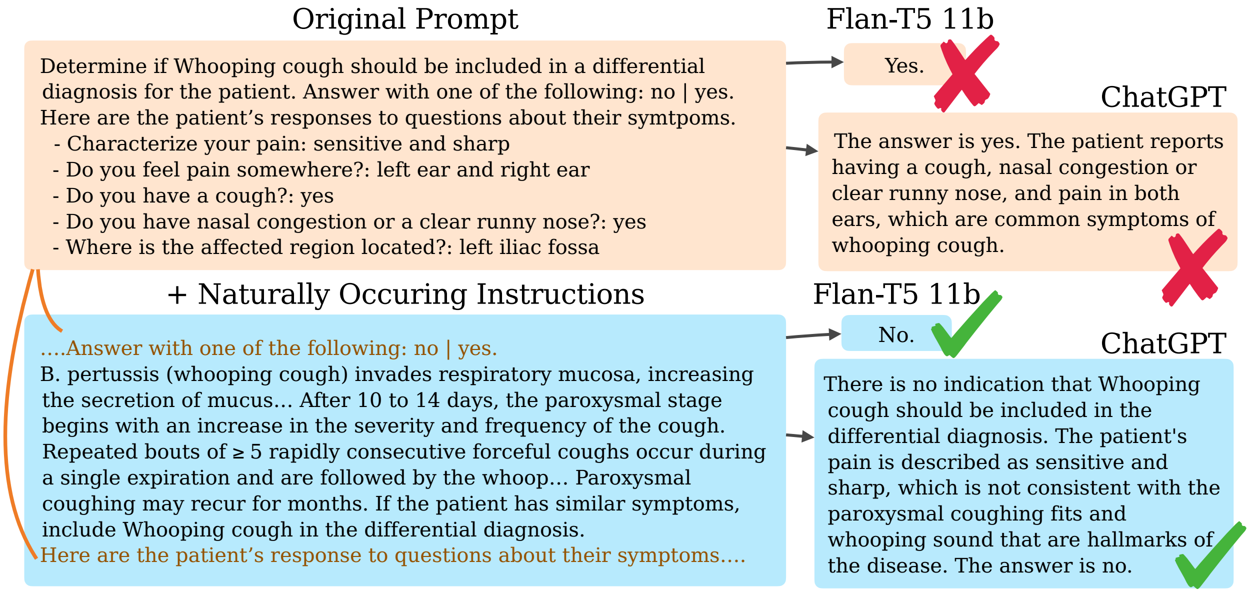
Explaining Machine Learning Models with Interactive Natural Language Conversations. He designed TalkToModel and produced all the results.

Chapter 7

TABLET: Natural Language Instructions for Tabular Prediction

In the previous chapter, we demonstrated how natural language is an ideal medium for interpreting and understanding the predictions of machine learning (ML) models. Could natural language also be useful for model development? In particular, could natural language help us leverage the extensive knowledge in large language models (LLMs) to create high-performance predictive models? In this chapter, we consider how natural language instructions can be useful for using LLMs to create performant classifiers for tabular data.

Tabular data plays a crucial role in solving machine learning (ML) problems across many critical domains like health care and finance. However, acquiring labeled data to train supervised learning models is often challenging. For example, there are privacy restrictions in finance and medicine that prevent data sharing [Hulsen, 2020]. Moreover, collecting data is very costly, imposing a significant financial barrier. These concerns make gathering training data highly challenging and sometimes impossible [Murdoch, 2021, Moorthy and Ghosh, 2015, Crawford and Schultz, 2014, Cath, 2018].



Determine if Whooping cough should be included in a differential diagnosis for the patient. Answer with one of the following: no | yes. Here are the patient's responses to questions about their symtpoms.
- Characterize your pain: sensitive and sharp
- Do you feel pain somewhere?: left ear and right ear
- Do you have a cough?: yes
- Do you have nasal congestion or a clear runny nose?: yes
- Where is the affected region located?: left iliac fossa

Yes. ✗

The answer is yes. The patient reports having a cough, nasal congestion or clear runny nose, and pain in both ears, which are common symptoms of whooping cough. ✗

....Answer with one of the following: no | yes.
B. pertussis (whooping cough) invades respiratory mucosa, increasing the secretion of mucus... After 10 to 14 days, the paroxysmal stage begins with an increase in the severity and frequency of the cough. Repeated bouts of ≥ 5 rapidly consecutive forceful coughs occur during a single expiration and are followed by the whoop... Paroxysmal coughing may recur for months. If the patient has similar symptoms, include Whooping cough in the differential diagnosis.
Here are the patient's response to questions about their symptoms....

No. ✓

There is no indication that Whooping cough should be included in the differential diagnosis. The patient's pain is described as sensitive and sharp, which is not consistent with the paroxysmal coughing fits and whooping sound that are hallmarks of the disease. The answer is no. ✓

Figure 7.1: **Evaluation with TABLET.** We serialize a short task description and a data point from the Whooping Cough dataset into a prompt (top). Flan-T5 11b and ChatGPT predict this instance incorrectly based on clues related to a cough and nasal congestion. However, by adding the instructions (bottom), the model identifies that the patient's pain is inconsistent with typical symptoms of Whooping Cough and correctly predicts the instance.

Large language models (LLMs) have the potential to considerably lighten the burden of collecting data for tabular prediction. Trained on web-scale data, LLMs have extensive world knowledge that is useful for downstream tasks [Brown et al., 2020]. Instead of gathering large datasets, practitioners could create models for tabular prediction by interactively providing *natural language instructions* to LLMs. For instance, a doctor knowledgeable about a rare disease could instruct an LLM to identify its symptoms. From there, the model could help identify the disease without sharing private training data or requiring data collection. Although using instructions for tabular prediction with LLMs has considerable potential, to what extent it would improve performance and its limitations are unclear.

To evaluate the performance of LLMs at learning from instructions for tabular prediction, we introduce a *TAbular Benchmark for LEarning from Task instructions* (TABLET). This benchmark consists of 20 diverse tabular prediction tasks. These include 10 datasets from the UCI ML repository, such as credit risk and churn prediction [Dua and Graff, 2017]. The additional 10 tasks are differential diagnosis (DDX) datasets for diseases. To robustly measure how effectively LLMs utilize instructions, we annotate the tasks with several generated and naturally occurring instructions categorized by their origin, structure, and level of technicality.

Using the instructions in TABLET, researchers can evaluate how well models perform at tabular prediction by learning *only* from in-context instructions (the *zero-shot* setting) or in the *few-shot* setting, where there are some labeled instances. Moreover, they can contrast performance between collection sources, e.g., consumer versus technical professional references, or evaluate how to best structure the prompts. We provide an example of evaluating with TABLET in Figure 7.1. When we provide LLMs with solely the original tabular data points serialized as text and a task description, LLMs fail to predict correctly. However, when we provide the instructions for identifying symptoms of Whooping Cough in TABLET, models consider more important features, such as paroxysmal coughing, and predict correctly. Our analysis on TABLET reveals zero-shot Flan-T5 11b improves F1 score by 20% on average over the same model without instructions, and ChatGPT F1 performance improves 10%. The performance gains increase in the few-shot setting; Flan-T5 11b improves on average 44% over the baseline with 4 in-context training examples, demonstrating instructions significantly improve LLM performance on tabular prediction.

To evaluate the limitations and failure modes of learning from instructions for tabular prediction, we include additional artifacts in TABLET, such as representations of the instruction's logic and *flipped* instructions (i.e., instructions that, if followed, should lead to different answers than the originals). In this setting, researchers can measure if models are faithful to instructions and use instructions for generalization benefits beyond just following the information in the instructions. While instructions improve LLM performance in general, we find LLMs are highly-biased against classifying certain instances correctly. Similarly, LLMs do not always follow instructions provided in-context. Overall, our analysis on TABLET demonstrates instructions are promising for improving LLM performance on tabular prediction tasks, but current models have several key limitations. In the future, we hope the release of TABLET enables researchers to develop models capable of solving tabular prediction tasks from instructions alone.

7.1 Tabular Instruction Learning

In this section, we introduce the problem of learning from instructions for tabular prediction.

7.1.1 Problem Formulation

This subsection formalizes our problem setting.

Tasks Each tabular prediction task t has dataset $D_t = \{(\mathbf{x}_i, y_i)\}_i^n$, where data points $\mathbf{x}_i$ have d features, the dataset D_t has n rows, and labels y_i belong to w classes $y_i \in C_t$. Additionally, the features and classes have names $F_t = \{f_1, ..., f_d\}$ and $C_t = \{c_1, ..., c_w\}$, respectively. We expect the feature names to be semantically meaningful, e.g., *education* or *marital status*, and the feature values to be in their normal range. Last, each task has a title E_t describing the goal (example in Table 7.1). It is usually possible to extract the feature names, class names, and title from the dataset's meta data Dua and Graff [2017].

Also, the dataset D_t splits into training and testing sets, D_t^{train} and D_t^{test} respectively. While the focus of TABLET is learning from instructions when there are *few* or *no* training examples in the training data D_t^{train}, we provide the full training sets in TABLET to compare against fully supervised models and perform few-shot evaluations.

E_t	Predict if income exceeds \$50K/yr.
I_t	Individuals who earn more than \$50K/yr tend to have higher education levels (e.g., Bachelors or Prof-school).
C_t	>50K | $\leq$ 50K
F_t	hours/week: 40 | sex: Female | age: 24 occupation: Sales | education: college

Table 7.1: **Example Adult instance** (abbreviated), with title E_t, instructions I_t, classes C_t, and features F_t.

Instructions Each task t has *natural language instructions* I_t that describe the relationship between the data D_t and labels C_t. The instructions may vary across many factors, such as style, granularity, and phrasing. The goal is to use the instructions to solve the task with few or no training examples.

7.1.2 Prompting Schema

We follow a consistent schema for collating tabular instruction learning problems into prompts, inspired by instruction learning for NLP Mishra et al. [2022], Wang et al. [2022]. For a task t, the prompts consist of five parts:

- **Title:** The description of the task E_t.
- **Instructions:** The instructions I_t.
- **Classes:** The classes C_t (e.g., "yes" or "no").
- **Examples:** In-context examples D_t^{train}, if any.
- **Test Data Point:** Test example, $\mathbf{x}_i \in D_t^{\text{test}}$.

To convert tabular data points into strings for the *Examples* and *Test Data Point* components, we follow Hegselmann et al. [2022]. We format feature names and values into strings:

`‘‘{f`$_1$`}:{x`$_1$`}\n...\n{f`$_d$`}:{x`$_d$`}’’`. We format the labels C_t into a string: `‘‘Answer with one of the following: {c`$_1$`}|...|{c`$_w$`}’’`.

7.2 TABLET

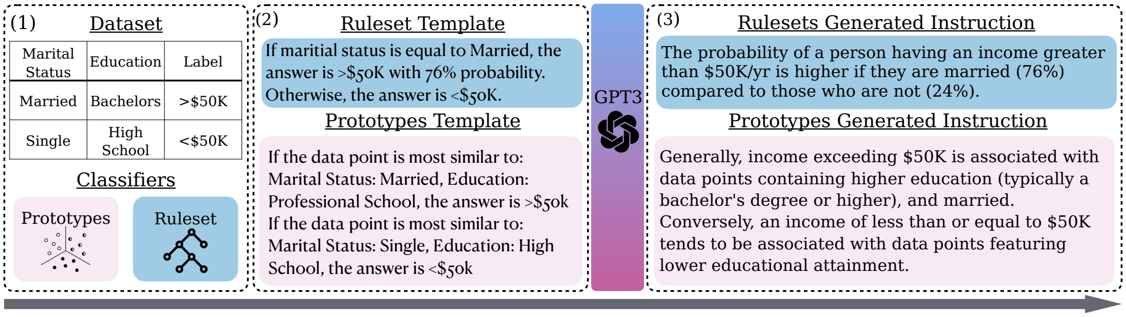
In this section, we introduce TABLET. The benchmark contains 20 tasks annotated with the components of the prompting schema (Subsection 7.1.2).

7.2.1 Tasks

Here, we describe the tasks in TABLET.

Differential Diagnosis We include 10 differential diagnosis (DDX) tasks derived from DDXPlus, a dataset for differential diagnosis carefully validated by doctors [Tchango et al., 2022]. These tasks aim to predict if a disease, such as Ebola or Myocarditis, could cause a patient's symptoms (example in Fig. 7.1). While the DDX problem is highly important, model development is limited by data privacy, making it an ideal task for instruction learning [Murdoch, 2021, Peng et al., 2018]. These tasks have sparse binary features, indicating whether a patient has a specific symptom. Because there are many features, we use only the positive features that indicate symptom presence in the prompts.

UCI We add 10 diverse tasks from the UCI ML Repository, such as Customer Churn and Credit [Dua and Graff, 2017]. These tasks represent many application areas and differ along dataset characteristics. For instance, this set includes datasets with numeric, categorical, and mixed features and different class sizes (binary to 16-way).



Marital Status	Education	Label
Married	Bachelors	>$50K
Single	High School	<$50K

If maritial status is equal to Married, the answer is >$50K with 76% probability. Otherwise, the answer is <$50K.

If the data point is most similar to: Marital Status: Married, Education: Professional School, the answer is >$50k If the data point is most similar to: Marital Status: Single, Education: High School, the answer is <$50k

The probability of a person having an income greater than $50K/yr is higher if they are married (76%) compared to those who are not (24%).

Generally, income exceeding $50K is associated with data points containing higher education (typically a bachelor's degree or higher), and married. Conversely, an income of less than or equal to $50K tends to be associated with data points featuring lower educational attainment.

Figure 7.2: **Instruction generation pipeline.** (1) We fit a prototypes and rulesets classifier on the dataset; here Adult (2) We serialize the classifier's logic into a template (3) We sample revisions of the template with GPT-3.

Splits We split the datasets with an 80/20 train/test split. While we include the full training data in TABLET, critically, our evaluation with LLMs takes place in the *zero* and *few-shot* settings. We include the full training set to permit comparison against fully supervised models and different few-shot samples. Because the DDX tasks have many records (> 1M), we down-sample to $10,000$ instances. The DDX data is highly imbalanced, so we rebalance the data before splitting and sample the test set in proportion to the original distribution.

7.2.2 Instructions

This section describes the instruction annotations.

Naturally Occurring Instructions

We collect 3 naturally occuring instruction annotations for each DDX dataset. Two of these instructions are from consumer friendly sources, while one is from a more technical professional reference. The consumer instructions are from government health websites and non-profits. To ensure quality, we use articles in an index of consumer medical information maintained by the National Library of Medicine called MedlinePlus [National Library of

Medicine (US)]. We collect the more technical instruction from a professional reference called the Merck Manual [Bullers, 2016]. To collect the instructions, first, we search for the disease using its ICD-10 code in either MedlinePlus or Merck. ICD-10 is a standardized disease classification system that helps us ensure we find the correct information [World Health Organization, 1993]. Next, we find the symptoms section on the webpage. We create the final instruction by combining the symptoms with a piece of text stating the disease should be in the differential diagnosis of patients with these symptoms (Fig. 7.1).

Generated Instructions

While we collect instructions from trustworthy sources for the DDX datasets, we introduce scalable and controllable techniques to generate instructions that vary in their phrasing, granularity and style for evaluating model robustness to different sorts of instructions on the UCI datasets.

Method To generate instructions (overview in Fig. 7.2), we initially fit a simple model, such as a shallow rule set, on the task's full training data (1). Next, we serialize the model's logic into text using a *template* (2). While the templates capture useful logic for solving the task, they are highly structured. We use GPT-3 (3) to revise the templates into natural language by prompting it to convert the templates into a concise paragraph [Brown et al., 2020]. We carefully review the generated instructions to ensure they are faithful to the template. Next, we discuss two models, *rule sets* and *prototypes classifiers*, we use to generate instructions.

Rule Sets Humans often express their understanding through logical rules [von Rueden et al., 2023, Xu et al., 2018, Hu et al., 2016]. Motivated by these findings, we use a rule sets classifier to create instruction templates. Rule sets consist of independent decision rules for predicting a class (Figure 7.2). Each decision rule contains an independent `if-then`

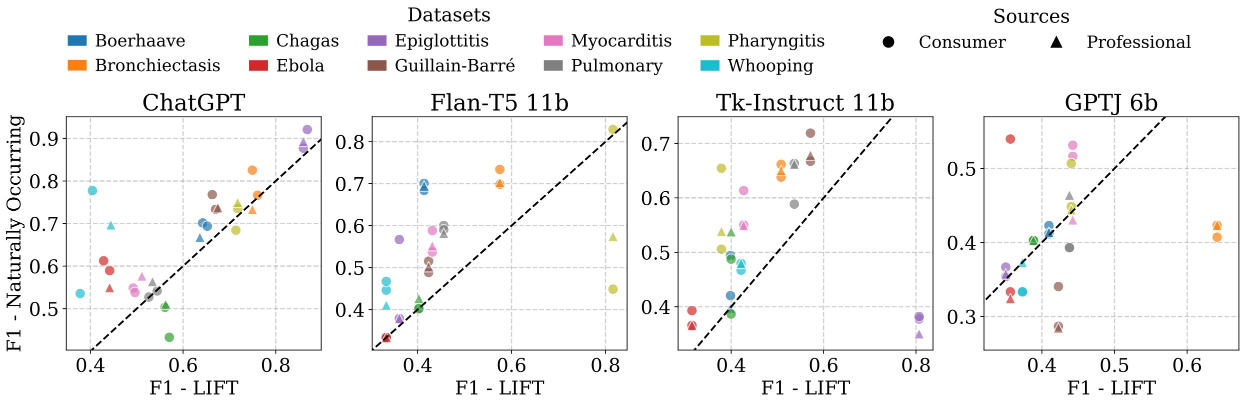


Figure 7.3: **Results on DDX tasks with Naturally Occurring Instructions.** Instructions greatly improve LLM generalization of ChatGPT, Flan-T5 11b, and Tk-Instruct 11b over the baseline without instructions (LIFT).

statement followed by a prediction and confidence value [Singh et al., 2021].

Prototypes Humans also represent knowledge using conceptual prototypes, representing the "best" examples of concepts [Osherson and Smith, 1981, Chen et al., 2019a]. We use a *prototypes* classifier as an instruction template. The prototypes classifier computes centroids for each class and uses the nearest centroid's class as the prediction (Figure 7.2).

Instructions We generate 10 prototypes and rulesets natural language instructions for the datasets in TABLET. To make the instructions more intelligible, we limit their complexity. We set the depth of the rulesets to 1. We select the 10 most important features for the prototypes classifier by computing their average information gain from XGBoost.

7.3 Experimental Setup

In this section, we describe our experiment setup.

Models We use pre-trained LLMs in our experiments Brown et al. [2020]. We employ two seq2seq models, Tk-Instruct 11b and Flan-T5 11b Wang et al. [2022], Chung et al. [2022]. These 11b T5 models received instruction fine-tuning, making them better at following instructions Raffel et al. [2020]. Also, we use GPT-J 6b, a causal language model, to compare a non-instruction tuned LLM Wang and Komatsuzaki [2021]. Last, we use ChatGPT, an LLM released by OpenAI trained with instruction tuning and reinforcement learning from human feedback (RLHF), a method that aligns LLMs with human preferences Schulman et al. [2022]. Because OpenAI provides ChatGPT access through a costly API, we use this model in select experiments.

Prompts To construct prompts, we use the templates provided by Tk-Instruct and Flan-T5. For example, Tk-Instruct prefers instructions to begin with `‘‘Definition:’’`. Because GPT-J and ChatGPT do not have templates, we create prompts by joining the annotations from the TABLET schema (subsection 7.1.2) with line breaks. For GPT-J, we append `‘‘The answer is’’` so the next token will be the class. For ChatGPT, we append `‘‘Please answer with` c_1`,..,or` c_w`.’’` because it is less amenable to direct completion.

Inference For all models except ChatGPT, we set the temperature to 0 and take the generation from the prompted LLM as the predicted class. However, ChatGPT often generates a paragraph of text instead of a label. So, we use nucleus sampling fixing $p = 0.1$. We take the label that occurs in the generation to be the prediction. However, label text may appear in other words, causing false positives (e.g., “yes” appears in “eyes”). So, we modified the class labels for ChatGPT to “The answer is c_i”, which did not occur by chance in our evaluation.

Additional Baselines & Metrics To benchmark against an approach without instructions on TABLET, we reproduce LIFT Dinh et al. [2022]. LIFT combines feature names, values,

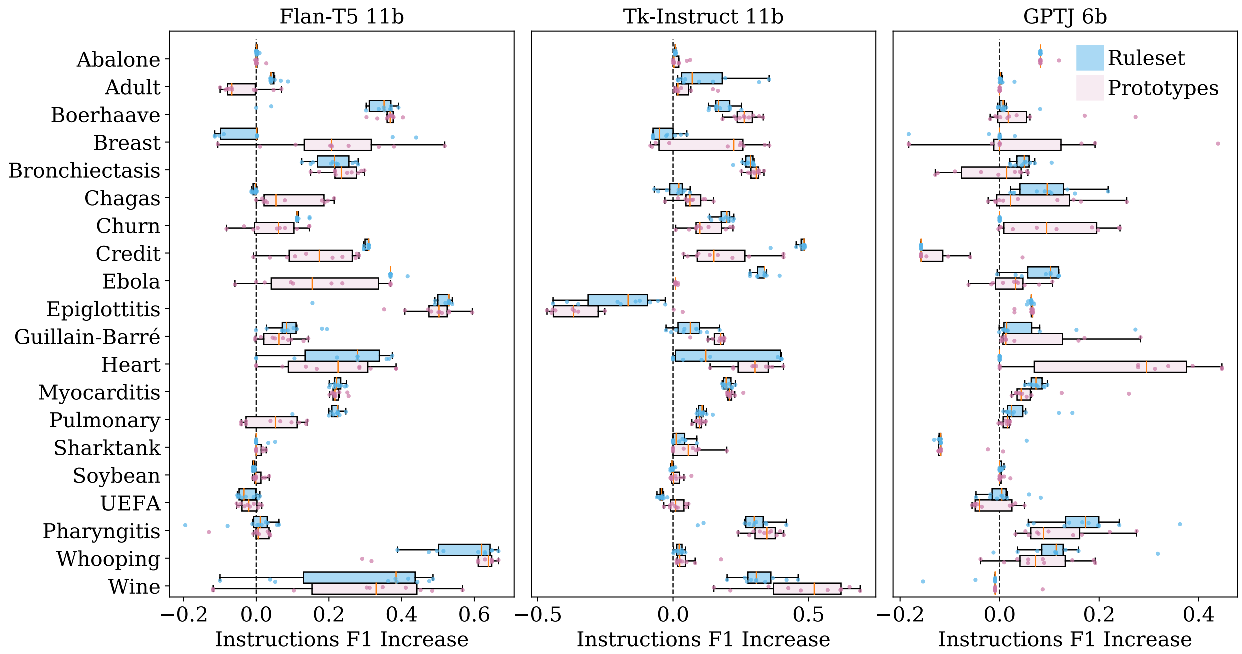


Figure 7.4: **Model results on all tasks with generated instructions.** Both prototypes and rulesets generated instructions lead to improved performance on most tasks for Flan-T5, Tk-Instruct, and GPT-J.

and a short description of the task into a prompt for in-context learning without instructions. We create prompts for LIFT from TABLET by omitting the instructions from the prompt. When we compare LLMs with instructions against LIFT, we use the *same* LLM, so the difference is the *presence of instructions*. Last, we evaluate using macro averaged F1 score for comparing across unbalanced labels.

7.4 Experiments

TABLET presents several new directions for evaluating how well instructions help LLMs use their knowledge for tabular prediction. First, we evaluate in the *zero-shot* setting and see whether LLMs perform well on tabular prediction with just instructions. Next, we consider the *few-shot* setting and assess LLM performance with both instructions and few-shot examples in-context. Throughout, we use TABLET to dissect the key limitations of current LLMs on tabular prediction tasks.

7.4.1 Zero-Shot Performance

Ideally, users could provide instructions to LLMs and achieve strong performance on tabular datasets without any data collection. Therefore, we initially explore the zero-shot setting.

Performance We compare the zero-shot F1 performance of LLMs using naturally occurring instructions with LIFT (no instructions) in Figure 7.3. We use a Wilcoxon signed rank test to evaluate if instructions lead to significantly better performance and correct p-values using Holm–Bonferroni to account for multiple comparisons on the datasets Holm [1979]. The naturally occurring instructions significantly improve performance for ChatGPT, Flan-T5 11b, and Tk-Instruct 11b (p¡0.01), while they are not helpful for GPT-J (p¿0.05). ChatGPT performs the best overall and scores quite highly for tasks such as Epiglottitis, reaching upwards of 0.9 F1. In Figure 7.4, we compare LLM performance using generated instructions with LIFT. We omit ChatGPT from these experiments due to costs. Generated instructions significantly increase performance for all models (p¡1e-4). The improved GPT-J 6b performance with generated instructions is likely because the instructions include exact terms from the dataset, like the feature names, making them easier to follow. Overall, instructions consistently benefit LLM performance on tabular prediction.

Differences Across Collection Sources We assess differences in performance on the consumer and professional references. For both Tk-Instruct 11b and Flan T5 11b, the instructions from the professional medical reference only performed best on 1/10 data sets (Fig. 7.3). However, the professional instructions performed best on 4/10 datasets for ChatGPT. These results could be explained by ChatGPT being a larger model capable of memorizing technical language during pre-training.

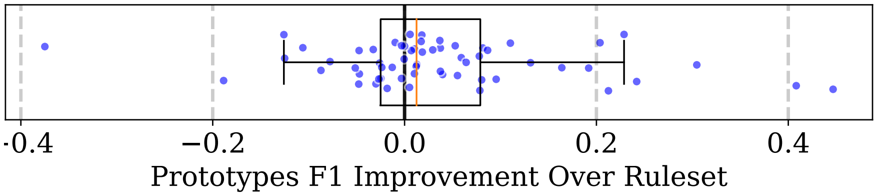


Figure 7.5: **Prototypes instructions perform better**.

Differences Across Generation Methods We contrast performance on the rule sets and prototypes generated instructions using GPT-J 6b, Tk-Instruct 11b, and Flan-T5 11b. Because the datasets have 10 annotations for both types of instructions, we take the max performance on the instruction type to compare the upper bound performance. The results provided in Figure 7.5 show that the prototypes instruction type perform better ($p = 0.03$). This result indicates LLMs have an easier time interpreting conceptual prototypes than logical rules.

Generalization Beyond Instruction Logic Recall we derived the generated instructions from supervised models (Subsection 7.2.2). We evaluate whether LLMs with these instructions generalize better than the original supervised models. Flan-T5 11b outperforms the prototypes classifiers by 33% on average across all the datasets and the rulesets classifiers by 91%. This result shows that LLMs leverage the instructions for generalization benefits beyond simply repeating the logic in the instruction.

Instruction Faithfulness We measure whether LLMs behave faithfully to instruction modifications. We create a *flipped* set of instructions by permuting the labels for the prototype class centroids and regenerating the instructions. LLM predictions should be different following the flipped instructions because each centroid has a different class label. We generate 10 flipped instructions for the Boerhaave, Breast Cancer, Epiglottis, Heart Disease, Wine, and Bronchiectasis datasets. We evaluate how often models predict the same class with the flipped instruction as the originals using Flan-T5 11b and Tk-Instruct 11b. We

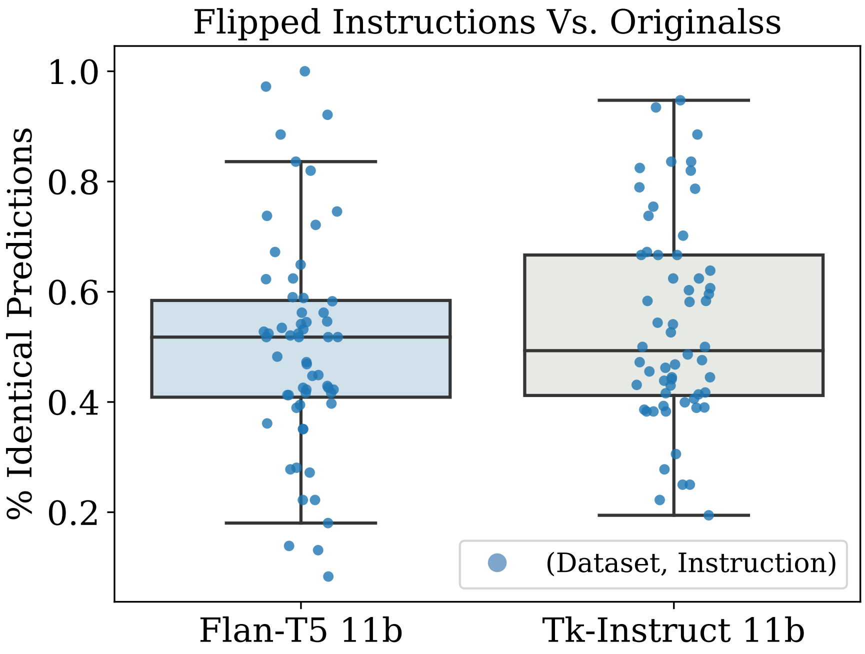


Figure 7.6: **LLMs predict many instances identically using instructions with *flipped* logic**, indicating they are overly biased.

present the results in Figure 7.6. Surprisingly, the median number of identical predictions is 52% for Flan and 49% for Tk-Instruct. This result demonstrates that LLMs may overly rely on their biases from pre-training instead of adapting to the in-context instructions.

7.4.2 Few-Shot Performance

Ideally, instructions alone would be sufficient to achieve good performance. However, few-shot examples could significantly help models solve tasks, while still alleviating the challenge of gathering a large training set. Therefore, this subsection studies using both instruction and few-shot examples for in-context learning. For these experiments, we evaluate few-shot performance on the DDX tasks for $[0, 2, 4]$ shots due to Flan's context window, using both LIFT and the naturally occurring instructions. We perform this evaluation with ChatGPT and Flan-T5 11b. To compare against state-of-the-art supervised models in this setting, we also evaluate few-shot XGBoost. Additionally, because of the cost of ChatGPT, we evaluate

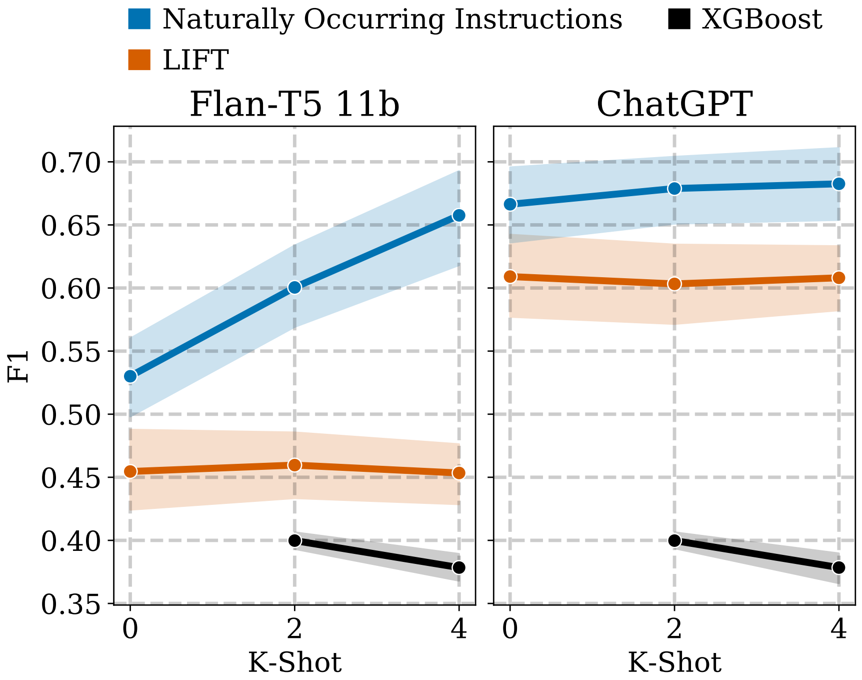


Figure 7.7: **Few-shot examples provide larger benefits to LLMs *with* instructions than *without* instructions.**

a single seed. Though, we perform few-shot sampling for 10 datasets, which reduces any bias. We present the results in Figure 7.7.

Performance Gap While zero-shot Flan-T5 with naturally occurring instructions outperforms LIFT by 20%, this gap increases to 44% with 4 in-context examples. Similarly, ChatGPT's performance gap increases from 10% to 13% with 4 examples. XGBoost scores only 0.38 F1 with 4-shot examples—much lower than the LLMs with instructions. These results show that few-shot examples have compounding benefits for models with instructions, and LLMs significantly outperform supervised models in the few-shot setting.

Instance Learnability Though the knowledge LLMs learn during pre-training helps performance, these models are overly biased and often fail to learn from few-shot examples. We compute the percent of the time Flan-T5 11b, with naturally occurring instructions and 4-

shot examples, predicts each test datapoint correctly, over 30 random seeds. Also, we compare against logistic regression resampling the dataset with replacement 30 times and perform this evaluation on the Ebola, Whooping cough, Bronchiectasis, and Epiglottitis datasets. The results in Figure 7.8 show Flan always mispredicts specific data points, indicating it is highly biased towards the opposite class label, and few-shot examples do little to help. Also, the logistic regression model struggles to learn fewer data points, indicating LLM biases is the leading cause of this result and not mislabeled or hard-to-learn examples. Overall, the knowledge in LLMs is quite helpful for performance, but these models are often inflexible and overly biased.

Comparison to Fully Supervised To evaluate how close LLMs are to outperforming fully supervised models, we compare Flan-T5 11b and ChatGPT with 4-shot examples, against XGBoost fit on the entire training data. XGBoost with all the data scores 0.94 F1 on average on the DDX tasks, while ChatGPT 4-shot scores on average 0.68 and Flan-T5 11b scores 0.66 F1. Thus, there is still considerable room for improvement on TABLET.

Few-Shot Selection We evaluate whether LLMs match fully supervised performance by selecting better few-shot examples instead of randomly sampling. Though this experiment assume access to the complete training set, which is not our intended use case, it provides insight into whether approximate upper bound LLM performance matches fully supervised. For each test point in the DDX tasks, we select 4 few-shot examples using feature-weighted K-Nearest Neighbors (KNN) on the original data. We compute feature weights using the average information gain of each feature from XGBoost fit on the training set. This KNN model scores 0.93 F1 on average on the tasks. However, Flan T5 11b scores 0.77 F1 using the same examples and naturally occurring instructions. Even in this ideal setting, Flan-T5 still underperforms fully supervised.

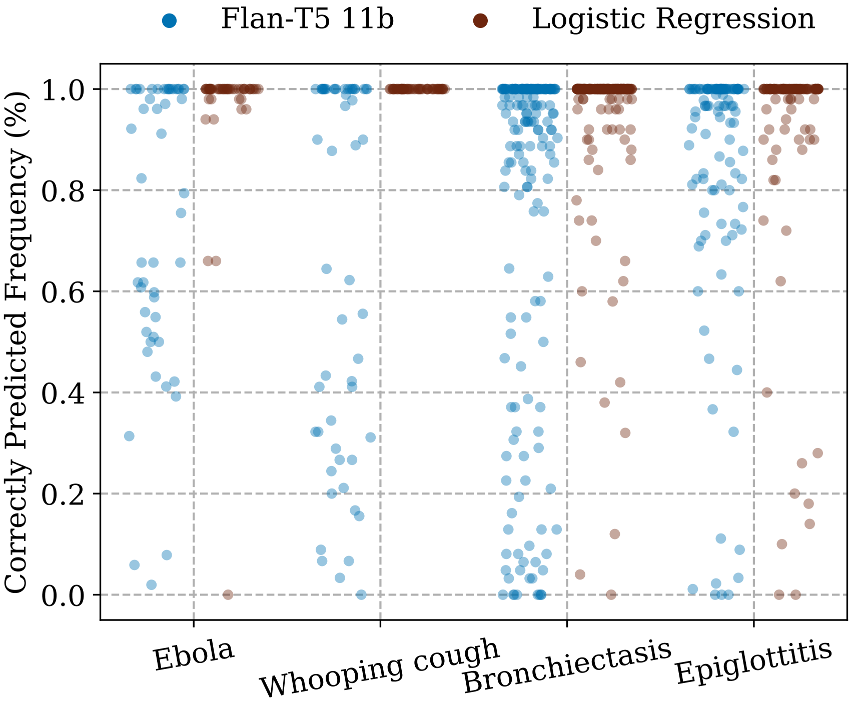


Figure 7.8: **LLMs + Instructions are highly biased against specific examples.** Each dot is a single data point. Over 30 seeds, LLMs + Instructions with 4-shot examples consistently misclassify specific data points.

7.5 Summary Of Contribution

In this chapter, we introduced TABLET, a new benchmark of tabular datasets annotated with instructions, where the goal is to use the instructions to solve the prediction problem. Improved methods for learning from the instructions for tabular prediction will significantly impact many domains. Indeed, tabular data is the most common data format for many fields that rely on relational databases, such as medicine, finance, and manufacturing [Shwartz-Ziv and Armon, 2021]. Thus, learning from instructions has the potential to reduce the need for costly and time-consuming data collection procedures in many areas, offering widespread benefits.

To support a wide range of possible use cases, we annotated the tasks in TABLET with instructions that vary along several factors, including their collection source, granularity, and phrasing. We presented results using several pre-trained LLMs, such as Flan-T5 11b and ChatGPT and found that instructions significantly increased task performance. Still, LLMs often do not follow instructions and struggle to predict specific instances correctly, even with in-context examples, demonstrating the limitations of current LLMs at this task. Accordingly, a promising direction for future research is developing models that strike a better balance between utilizing knowledge from pre-training when it is helpful to the task and faithfully adapting to new information.

This chapter is based on the following manuscript:

- TABLET: Learning From Instructions For Predicting Tabular Data With Language Models. (Slack and Singh [2023], arXiv)

The author of this dissertation is responsible for all the results in this chapter.

Chapter 8

Conclusions and Future Work

First, this dissertation evaluated the limitations of machine learning explanations and developed more robust explanation techniques. We showed that broadly used types of explanations, including post hoc and counterfactual explanations, are not robust to manipulation by adversaries. These results raise alarming concerns about the reliability of explanations, and also serve to highlight the key limitations of these techniques, such as the perturbation distribution in LIME. Moreover, we developed Bayesian explanation methods that reduced several critical robustness issues of post hoc explanations, such as instability and inconsistency, through improved uncertainty modeling.

Second, this dissertation proposed *natural language* as a better way for humans to interact with ML models and demonstrated how it can be used for improved explainability and model development with TalkToModel and TABLET, respectively. Our results evaluating with TalkToModel showed humans strongly preferred our conversational system for explaining machine learning models over traditional explanainability systems, such as point and click dashboards for interfacing with LIME and SHAP. Similarly, we found natural language instructions can utilize the extensive knowledge in LLMs for improving predictive performing on tabular datasets, without requiring and training data collection. Taken together, these results show natural language is a highly effective way for performing several different aspects of the model develop process, such as training and understanding model predictions, providing the groundwork for further investigation into different uses of natural language for model development.

8.1 Impact

The contributions in this dissertation offer several routes for understanding the limitations of explanation methods, which is critical for researchers developing improved techniques, and provides several new explainability methods. Moreover, this dissertation shows how natural language is highly useful for interacting with and using machine learning models, opening up many novel research problems. These ideas have already had meaningful impact on related works; for instance, the papers covered in this dissertation have over 622 citations according to Google Scholar. In particular, the vulnerabilities we demonstrated in feature importance and counterfactual explanations have inspired many efforts to develop more robust explanation methods [Artelt and Hammer, 2022, Alvarez and Menzies, 2023, Carmichael and Scheirer, 2022, Vadillo et al., 2021, Vrevs and vSikonja, 2021]. Relatedly, TalkToModel and, more generally, our work on using natural language as a better route for model explainability have influenced several following works [Majumder et al., 2022, Hartmann et al., 2022, Feldhus et al., 2022a], and we expect that as LLMs continue to become more capable tools for general purpose use, the popularity of these techniques to grow.

8.2 Future Work

In this section, we consider promising avenues for future work, based on the findings of this dissertation.

Evaluating Faithfulness of Language Model Explanations In Chapter 3, we discussed how post hoc explanations might not be faithful to the model and can be designed to be misleading. Along the same lines as these vulnerabilities, it would be interesting to evaluate whether LLM explanations are ever misleading. In particular, while post hoc explanations are commonly used for tabular data prediction, LLMs are capable of rationalizing their own generations, and these rationalizations can be taken as explanations for LLM predictions on tasks. Therefore, it would be very useful to understand how grounded LLM explanations are to the internal reasoning process of the model. For instance, do LLMs ever provide explanations that are satisfactory to humans but are not reflective of how they actually solved the problem? It would additionally be useful to investigate how to guarantee that LLM explanations are aligned with their reasoning process. Such guarantees would ensure that LLM rationalizations are trustworthy and increase their usability.

LLMs as General Purpose Explainability Tools In the first part of this dissertation, we discussed several widely used explanations, such as LIME and SHAP. While these methods help explain ML models, they ultimately use simple models like regressions to capture the model's behavior in a small region on the decision boundary of a model. Thus, because the explanation models are pretty simple, they might not fully capture the full nuance of the model's decision boundary. Moreover, users of the explanations risk misinterpreting the simple model if they are not used to working with ML, for instance. Therefore, a promising direction is creating ways to produce textual explanations of machine learning models using LLMs. For example, can LLMs generate a concise, natural language explanation for a model's prediction based on a series of perturbations, like the perturbations from LIME and SHAP?

Beyond Parsing based Explainability Dialogue Systems While we found TalkToModel was quite successful at engaging in explainability conversations using the parsing-based system, the system could become more flexible if it could directly write the code to run explanations on ML models and generate custom text responses. This functionality could enable the system to perform a broader range of analyses because it does not have to be explicitly coded into the parsing system. These directions are highly promising since LLMs have made considerable strides at generating code from text descriptions and generating useful text [OpenAI, 2021].

Building Classifiers Interactively with LLMs While our work with TABLET showed that it is possible to build high-performance classifiers using LLMs and natural language instructions, it would be helpful to investigate how to make this technique best usable in practice. In particular, it could be better for the LLM to support interactively receiving instructions and verifying understanding to the instruction provider. In a high-stakes scenario such as medicine, for instance, its likely users would wish to test the system's understanding before it is deployed to ensure it makes good predictions and try to correct faulty reasoning beforehand. Therefore, another promising direction for future research is developing interactive ways of prompting LLMs for tabular prediction.

Bibliography

Catalogue of standard toxicity tests for ecological risk assessment. *United States Environment Protection Agency*, 1994.

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9525–9536, Red Hook, NY, USA, 2018. Curran Associates Inc.

C. Agarwal, N. Johnson, S. K. Martin Pawelczyk, E. Saxena, M. Zitnik, and H. Lakkaraju. Rethinking stability for attribution-based explanations. *arXiv*, 2022a.

C. Agarwal, S. Krishna, E. Saxena, M. Pawelczyk, N. Johnson, I. Puri, M. Zitnik, and H. Lakkaraju. OpenXAI: Towards a transparent evaluation of model explanations. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022b.

R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton. Neural additive models: Interpretable machine learning with neural nets. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4699–4711. Curran Associates, Inc., 2021a.

S. Agarwal, S. Jabbari, C. Agarwal, S. Upadhyay, S. Wu, and H. Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 110–119. PMLR, 18–24 Jul 2021b.

L. Alvarez and T. Menzies. Don't lie to me: Avoiding malicious explanations with stealth. *arXiv*, 2023.

D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *ICML Workshop on Human Interpretability in Machine Learning*, 2018.

J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias. In *ProPublica*, 2016.

A. Artelt and B. Hammer. Explain it in the same way! - model-agnostic group fairness of counterfactual explanations. *arXiv*, 2022.

S. Barocas, A. D. Selbst, and M. Raghavan. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 80–89, New York, NY, USA, 2020. Association for Computing Machinery.

D. Bertsekas and R. Gallager. *Data Networks (2nd Ed.).* Prentice-Hall, Inc., USA, 1992.

U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. F. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 648–657, New York, NY, USA, 2020. Association for Computing Machinery.

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

C. Blake, E. Koegh, and C. Mertz. Repository of machine learning. *University of California at Irvine*, 1999.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

K. Bullers. Merck manuals. *Journal of the Medical Library Association : JMLA*, 104(4): 369–371, 2016.

W. Cai and L. Chen. Predicting user intents and satisfaction with dialogue-based conversational recommendations. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20, page 33–42, New York, NY, USA, 2020. Association for Computing Machinery.

G. Carenini, V. O. Mittal, and J. D. Moore. Generating patient-specific interactive natural language explanations. *Proc Annu Symp Comput Appl Med Care*, pages 5–9, 1994.

Z. Carmichael and W. J. Scheirer. Unfooling perturbation-based post hoc explainers. *arXiv*, 2022.

C. Cath. Governing artificial intelligence: ethical, legal and technical opportunities and challenges. *Philos Trans A Math Phys Eng Sci*, 376(2133), Oct. 2018.

C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin. *This Looks like That: Deep Learning for Interpretable Image Recognition.* Curran Associates Inc., Red Hook, NY, USA, 2019a.

H. Chen, X. Liu, D. Yin, and J. Tang. A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35, nov 2017.

J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations*, 2019b.

Q. Chen, T. Schnabel, B. Nushi, and S. Amershi. Hint: Integration testing for ai-based features with humans in the loop. In *International Conference on Intelligent User Interfaces*. ACM, March 2022.

H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models. arXiv, 2022.

K. Crawford and J. Schultz. Big data and due process: Toward a framework to redress predictive privacy harms. *Boston College Law Review*, 55(1), January 2014.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 1269–1277, Cambridge, MA, USA, 2014. MIT Press.

A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 592–603. Curran Associates, Inc., 2018.

O. Dijk, oegesam, R. Bell, Lily, Simon-Free, B. Serna, rajgupt, yanhong-zhao ef, A. Gädke, Hugo, and T. Okumus. oegedijk/explainerdashboard: v0.3.8.2: reverses set_shap_values bug introduced in 0.3.8.1. Apr. 2022. doi: 10.5281/zenodo.6408776. URL `https://doi.org/10.5281/zenodo.6408776`.

T. Dinh, Y. Zeng, R. Zhang, Z. Lin, M. Gira, S. Rajput, J.-y. Sohn, D. Papailiopoulos, and K. Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11763–11784. Curran Associates, Inc., 2022.

A.-K. Dombrowski, M. Alber, C. J. Anders, M. Ackermann, K.-R. Müller, and P. Kessel. Explanations can be manipulated and geometry is to blame. *arXiv*, 2019.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.

J. Dressel and H. Farid. The accuracy, fairness, and limits of predicting recidivism. *Science Advances*, 4(1), 2018.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

R. Elshawi, M. H. Al-Mallah, and S. Sakr. On the interpretability of machine learning-based model for predicting hypertension. *BMC medical informatics and decision making*, 19(1): 146, 2019.

L. Fahrmeir, T. Kneib, and S. Lang. *Regression.* Statistik und ihre Anwendungen. Springer, Berlin [u.a.], 2007.

N. Feldhus, A. Ravichandran, and S. Möller. Mediators: Conversational agents explaining nlp model behavior. *IJCAI-ECAI 2022 Workshop on XAI*, 06 2022a.

N. Feldhus, A. M. Ravichandran, and S. Möller. Mediators: Conversational agents explaining nlp model behavior. *arXiv*, 2022b.

B. Fernando and S. Gould. Learning end-to-end video classification with rank-pooling. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1187–1196, New York, New York, USA, 20–22 Jun 2016. PMLR.

M. Freed, J. Carbonell, G. Gordon, J. Hayes, B. A. Myers, D. Siewiorek, S. Smith, A. Steinfeld, and A. Tomasic. Radar: A personal assistant that learns to reduce email overload. In *Proceedings of 23rd National Conference on Artificial Intelligence*. AAAI, 2008.

S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, and D. Roth. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 329–338, New York, NY, USA, 2019. Association for Computing Machinery.

J. Gao, M. Galley, and L. Li. Neural approaches to conversational AI. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 2–7, Melbourne, Australia, July 2018. Association for Computational Linguistics.

S. Gao, A. Sethi, S. Agarwal, T. Chung, and D. Hakkani-Tur. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273, Stockholm, Sweden, Sept. 2019. Association for Computational Linguistics.

A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

A. Glass, D. L. McGuinness, and M. Wolverton. Toward establishing trust in adaptive agents. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, page 227–236, New York, NY, USA, 2008. Association for Computing Machinery.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

S. Gould, B. Fernando, A. Cherian, P. Anderson, R. Santa Cruz, and E. Guo. On Differentiating Parameterized Argmin and Argmax Problems with Application to Bi-level Optimization. *arXiv*, 2016.

B. M. Greenwell, B. C. Boehmke, and A. J. McCarthy. A simple and effective model-based variable importance measure. *arXiv*, 2018.

B. J. Grosz, A. K. Joshi, and S. Weinstein. Providing a unified account of definite noun phrases in discourse. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 44–50, Cambridge, Massachusetts, USA, June 1983. Association for Computational Linguistics.

D. Guo, D. Tang, N. Duan, M. Zhou, and J. Yin. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

S. Gupta, S. Singh, and M. Gardner. Structurally diverse sampling reduces spurious correlations in semantic parsing datasets. *arXiv*, 2022.

V. Gupta, P. Nokhiz, C. Dutta Roy, and S. Venkatasubramanian. Equalizing recourse across groups. *arXiv*, 2019.

M. Hartmann, H. Du, N. Feldhus, I. Kruijff-Korbayová, and D. Sonntag. Xaines: Explaining ai with narratives. *KI - Künstliche Intelligenz*, 36(3):287–296, 2022.

S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag. Tabllm: Few-shot classification of tabular data with large language models. *arXiv*, 2022.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. Accessed 1 Apr. 2023.

S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim. *A Benchmark for Interpretability Methods in Deep Neural Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

T. Hulsen. Sharing is Caring-Data sharing initiatives in healthcare. *Int J Environ Res Public Health*, 17(9), Apr. 2020.

M. Ibrahim, M. Louie, C. Modarres, and J. Paisley. Global explanations of neural networks: Mapping the landscape of predictions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, pages 279–287, 2019.

M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *BMVC 2014 - Proceedings of the British Machine Vision Conference 2014*, 2014.

A.-H. Karimi, G. Barthe, B. Balle, and I. Valera. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pages 895–905. PMLR, August 2020. URL `http://proceedings.mlr.press/v108/karimi20a.html`.

A.-H. Karimi, G. Barthe, B. Schölkopf, and I. Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv e-prints*, art. arXiv:2010.04050, October 2020.

B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. S ayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

S. Krishna, T. Han, A. Gu, J. Pombra, S. Jabbari, S. Wu, and H. Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. arXiv, 2022.

H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 131–138, New York, NY, USA, 2019. Association for Computing Machinery.

H. Lakkaraju, D. Slack, Y. Chen, C. Tan, and S. Singh. Rethinking explainability as a dialogue: A practitioner's perspective. *HCAI @ NeurIPS*, 2022.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

E. Lee, D. Braines, M. Stiffler, A. Hudler, and D. Harborne. Developing the sensitivity of lime for better machine learning explanation. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100610. International Society for Optics and Photonics, 2019.

J. Li, V. Nagarajan, G. Plumb, and A. Talwalkar. A learning theoretic perspective on local explainability. In *International Conference on Learning Representations*, 2021.

X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz. End-to-end task-completion neural dialogue systems. In *the 8th International Joint Conference on Natural Language Processing*. IJCNLP 2017, November 2017.

Q. V. Liao, D. Gruen, and S. Miller. Questioning the ai: Informing design practices for explainable ai user experiences. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.

Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57, June 2018.

Y. Liu, K. Han, Z. Tan, and Y. Lei. Using context information for dialog act classification in DNN framework. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Neural Information Processing Systems (NIPS)*, pages 4765–4774. Curran Associates, Inc., 2017a.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017b.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017c.

S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, Jan 2020.

B. P. Majumder, Z. He, and J. McAuley. Interfair: Debiasing with natural language feedback for fair interpretable predictions. *arXiv*, abs/2210.07440, 2022.

P. McGarry. Performance incentive funding. In *VERA*, 2012.

C. Meng, L. Trinh, N. Xu, J. Enouen, and Y. Liu. Interpretability and fairness evaluation of deep learning models on mimic-iv dataset. *Scientific Reports*, 12(1):7166, May 2022.

S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics.

B. Mittelstadt, C. Russell, and S. Wachter. Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 279–288. ACM, 2019.

A. Moore. Locally weighted bayesian regression, January 1995.

J. Moorthy and P. Ghosh. Big data and consumer privacy. *Vikalpa: The Journal for Decision Makers*, 40:92 – 95, 03 2015.

R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617, 2020.

B. Murdoch. Privacy and artificial intelligence: challenges for protecting health information in a new era. *BMC Medical Ethics*, 22(1):122, Sep 2021.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

National Library of Medicine (US). Medlineplus. [updated Jun 24; cited 2020 Jul 1]. URL `https://medlineplus.gov/`.

OpenAI. ChatGPT [computer software], 2021. URL `https://openai.com/blog/chat-gpt-3-app/`.

I. Oren, J. Herzig, N. Gupta, M. Gardner, and J. Berant. Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online, Nov. 2020. Association for Computational Linguistics.

D. N. Osherson and E. E. Smith. On the adequacy of prototype theory as a theory of concepts. *Cognition*, 9(1):35–58, 1981.

T. Paek and E. Horvitz. Conversation as action under uncertainty. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, page 455–464, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

S. Palan and C. Schitter. Prolific.ac—a subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, 17:22–27, 2018.

M. Pawelczyk, K. Broelemann, and G. Kasneci. *Learning Model-Agnostic Counterfactual Explanations for Tabular Data*, page 3126–3132. Association for Computing Machinery, New York, NY, USA, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Y.-S. Peng, K.-F. Tang, H.-T. Lin, and E. Chang. Refuel: Exploring sparse features in deep reinforcement learning for fast disease diagnosis. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

J. W. Pennebaker, M. R. Mehl, and K. G. Niederhoffer. Psychological aspects of natural language. use: our words, our selves. *Annu Rev Psychol*, 54:547–577, June 2002.

G. Plumb, D. Molitor, and A. Talwalkar. Model agnostic supervised local explanations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2520–2529, Red Hook, NY, USA, 2018. Curran Associates Inc.

R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach. Face: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 344–350, New York, NY, USA, 2020. Association for Computing Machinery.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

M. Redmond and A. Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3): 660–678, 2002.

G. D. P. Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88):294, 2016.

N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

M. T. Ribeiro, S. Singh, and C. Guestrin. Model-Agnostic Interpretability of Machine Learning. In *ICML Workshop on Human Interpretability in Machine Learning (WHI)*, June 2016a.

M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*, 2016b.

V. Rieser and O. Lemon. *Developing Dialogue Managers from Limited Amounts of Data*, pages 5–17. Springer New York, New York, NY, 2012.

C. Rudin, C. Wang, and B. Coker. The age of secrecy and unfairness in recidivism prediction. *Harvard Data Science Review*, 2(1), 3 2020.

S. Santhanam and S. Shaikh. A survey of natural language generation techniques with a focus on dialogue systems - past, present and future directions. *arXiv*, 2019.

J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng, J. F. C. Uribe, L. Fedus, L. Metz, M. Pokorny, R. G. Lopes, S. Zhao, A. Vijayvergiya, E. Sigler, A. Perelman, C. Voss, M. Heaton, J. Parish, D. Cummings, R. Nayak, V. Balcom, D. Schnurr, T. Kaftan, C. Hallacy, N. Turley, N. Deutsch, V. Goel, J. Ward, A. Konstantinidis, W. Zaremba, L. Ouyang, L. Bogdonoff, J. Gross, D. Medina, S. Yoo, T. Lee, R. Lowe, D. Mossing, J. Huizinga, R. Jiang, C. Wainwright, D. Almeida, S. Lin, M. Zhang, K. Xiao, K. Slama, S. Bills, A. Gray, J. Leike, J. Pachocki, P. Tillet, S. Jain, G. Brockman, N. Ryder, A. Paino, Q. Yuan, C. Winter, B. Wang, M. Bavarian, I. Babuschkin, S. Sidor, I. Kanitscheider, M. Pavlov, M. Plappert, N. Tezak, H. Jun, W. Zhuk, V. Pong, L. Kaiser, J. Tworek, A. Carr L. Weng, S. Agarwal, K. Cobbe, V. Kosaraju, A. Power, S. Polu, J. Han, R. Puri, S. Jain, B. Chess, C. Gibson, O. Boiko, E. Parparita, A. Tootoonchian, K. Kosic, and C. Hesse. Introducing chatgpt, Nov 2022.

P. Schwab and W. Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10220–10230. Curran Associates, Inc., 2019.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

B. Settles. Active learning literature survey. 2010.

Y. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

S. Sharma, J. Henderson, and J. Ghosh. Certifai: A common framework to provide explanations and analyse the fairness and robustness of black-box models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 166–172, New York, NY, USA, 2020. Association for Computing Machinery.

R. Shin, C. H. Lin, S. Thomson, C. Chen, S. Roy, E. A. Platanios, A. Pauls, D. Klein, J. Eisner, and B. V. Durme. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Nov. 2021.

R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

C. Singh, K. Nasseri, Y. S. Tan, T. Tang, and B. Yu. imodels: a python package for fitting interpretable models, 5 2021. URL `https://doi.org/10.21105/joss.03192`.

D. Slack and S. Singh. Tablet: Learning from instructions for predicting tabular data with language models. *arXiv*, 2023.

D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES)*, 2020.

D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9391–9404. Curran Associates, Inc., 2021a.

D. Slack, S. Hilgard, H. Lakkaraju, and S. Singh. Counterfactual Explanations Can Be Manipulated. In *Neural Information Processing Systems (NeurIPS)*, 2021b.

D. Slack, S. Krishna, H. Lakkaraju, and S. Singh. Talktomodel: Explaining machine learning models with interactive natural language conversations. *TSRML @ NeurIPS*, 2022.

K. Sokol and P. A. Flach. Glass-box: Explaining ai decisions with counterfactual statements through conversation with a voice-enabled virtual assistant. In *IJCAI*, pages 5868–5870, 2018.

K. Sokol, A. Hepburn, R. Santos-Rodríguez, and P. A. Flach. blimey: Surrogate prediction explanations beyond lime. *NeurIPS HCML Workshop*, 2019.

I. Stepin, J. M. Alonso, A. Catala, and M. Pereira-Fariña. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001, 2021.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

A. Talmor, M. Geva, and J. Berant. Evaluating semantic parsing against a simple web-based question answering model. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 161–167, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics.

H. F. Tan, K. Song, M. Udell, Y. Sun, and Y. Zhang. "why should you trust my explanation?" understanding uncertainty in lime explanations. In *ICML Workshop on AI for Social Good*, 2019.

S. Tan, R. Caruana, G. Hooker, and Y. Lou. Distill-and-compare: auditing black-box models using transparent model distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 303–310. ACM, 2018.

A. F. Tchango, R. Goel, Z. Wen, J. Martel, and J. Ghosn. DDXPlus: A new dataset for automatic medical diagnosis. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

B.-H. Tseng, S. Bhargava, J. Lu, J. R. A. Moniz, D. Piraviperumal, L. Li, and H. Yu. CREAD: Combined resolution of ellipses and anaphora in dialogues. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3390–3406, Online, June 2021. Association for Computational Linguistics.

B. Ustun, A. Spangher, and Y. Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 10–19. ACM, 2019.

J. Vadillo, R. Santana, and J. A. Lozano. When and how to fool explainable models (and humans) with adversarial examples. *arXiv*, 2021.

A. Van Looveren and J. Klaise. Interpretable Counterfactual Explanations Guided by Prototypes. *arXiv*, 2019.

S. Venkatasubramanian and M. Alfano. The philosophical basis of algorithmic recourse. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 284–293, New York, NY, USA, 2020. Association for Computing Machinery.

S. Verma, J. Dickerson, and K. Hines. Counterfactual explanations for machine learning: A review. arXiv, 2020.

L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage, and J. Schuecker. Informed machine learning – a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2023.

D. Vrevs and M. R. vSikonja. Better sampling in explanation methods can prevent dieselgate-like deception. 2021.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard journal of law & technology*, 31: 841–887, 2018.

B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, et al. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*, 2022.

N. G. Ward and D. DeVault. Challenges in building highly-interactive dialog systems. *AI Magazine*, 37(4):7–18, Jan. 2017.

L. S. Whitmore, A. George, and C. M. Hudson. Mapping chemical performance on molecular structures using locally interpretable explanations. *arXiv*, 2016.

World Health Organization. *The ICD-10 Classification of Mental and Behavioural Disorders*. World Health Organization, 1993.

S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022.

J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5502–5511. PMLR, 10–15 Jul 2018.

P. Yin, H. Fang, G. Neubig, A. Pauls, E. A. Platanios, Y. Su, S. Thomson, and J. Andreas. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online, June 2021. Association for Computational Linguistics.

T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018. Association for Computational Linguistics.

Z. Zhang, R. Takanobu, Q. Zhu, M. Huang, and X. Zhu. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63(10):2011–2027, Oct. 2020.

Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 18–24 Jul 2021.

Appendix A

Reliable Post hoc Explanations

A.1 Additional Derivations

Model Derivation We write the joint posterior as

$$\phi, \sigma^2 | Y, \mathcal{Z} \propto \rho(Y|X, \beta, \sigma^2)\rho(\beta|\sigma^2)\rho(\sigma^2) \tag{A.1}$$

$$\begin{aligned} &\propto (\sigma^2)^{-N/2}\text{exp}(-\frac{1}{2\sigma^2}(Y - \mathcal{Z}\phi)^T \text{diag}(\Pi_x(\mathcal{Z}))\cdot \\ &(Y - \mathcal{Z}\phi))(\sigma^2)^{-1}\text{exp}(-\frac{1}{2\sigma^2}\phi^T\phi)(\sigma^2)^{-(1+\frac{n_0}{2})}\text{exp}\left[\frac{-n_0\sigma_0^2}{2\sigma^2}\right] \end{aligned} \tag{A.2}$$

Letting $\hat{\phi} = (\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))Y$, we group terms in the exponentials according to ϕ. The intermediate steps can be found in Fahrmeir et al. [2007]. Supressing dependence on Y and $\mathcal{Z}$, we can write down the conditional posterior of ϕ as

$$\phi|\sigma^2 \propto \text{exp}(\frac{1}{2}\sigma^{-2}[\phi - \hat{\phi}]^T(\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)[\phi - \hat{\phi}]) \tag{A.3}$$

So, we can see that our estimates for the mean and variance of $\rho(\phi|\sigma^2, Y, \mathcal{Z})$ are $\hat{\phi}$ and $\sigma^2(\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}$. Next, we derive the conditional posterior for σ^2. We identify the form of the scaled inverse-χ^2 distribution in the joint posterior as in Moore [1995] and write

$$\sigma^2|\hat{\phi} \sim \text{Inv-}\chi^2(N+n_0, \frac{n_0\sigma_0^2 + Ns^2}{n_0+N}) \tag{A.4}$$

where s^2 is defined as in equation 5.5.

Derivation of equation 5.6 We establish the identity Moore [1995]:

$$\begin{aligned} \sigma^2 \sim \text{Inv-}\chi^2(a,b) \text{ and } z|\sigma^2 \sim \mathcal{N}(\mu, \lambda\sigma^2) \\ \iff z \sim t_{(\mathcal{V}=a)}(\mu, \lambda b) \end{aligned} \tag{A.5}$$

We have, $\epsilon \sim \mathcal{N}(0,\sigma^2)$, $\sigma^2 \sim \text{Inv-}\chi^2(N+n_0, \frac{n_0\sigma_0^2+Ns^2}{n_0+N})$. Then, it's the case that $\epsilon \sim t_{(\mathcal{V}=N+n_0)}(0, \frac{n_0\sigma_0^2+Ns^2}{n_0+N})$.

Derivation of Posterior Predictive Note, this derivation takes the priors to be set as in BayesLIME or BayesSHAP, namely, with values close to zero. We apply the identity from equation A.5 to derive this posterior. We have $\hat{y} \sim \hat{\phi}^T z + \epsilon$ for some z. Thus, $\hat{y} \sim \mathcal{N}(\hat{\phi}^T z, z^T V_\phi z \sigma^2) + \mathcal{N}(0, \sigma^2)$, where $\sigma^2 \sim \text{Inv-}\chi^2(N, s^2)$. So, we have $\hat{y} \sim t_{(\mathcal{V}=N)}(\hat{\phi}^T z, (z^T V_\phi z + 1)s^2)$.

A.2 Proof of Theorems

In these derivations, the perturbation matrices $\mathcal{Z}$ have elements $\mathcal{Z}_{ij} \in \{0,1\}$ where each $\mathcal{Z}_{ij} \sim \text{Bernoulli}(0.5)$. Note, in these proofs, we take take the priors to be set as in BayesLIME and BayesSHAP, i.e., they have hyperparameter values close to 0.

A.2.1 Proof of Theorem 5.1

Note that we use N to denote the *total* perturbations while S denotes the perturabtions collected *so far*. We use three assumptions stated as follows. First, $\frac{\bar{\pi}N}{2}$ is sufficiently large such at $\frac{\bar{\pi}N}{2}+1$ is equivalent to $\frac{\bar{\pi}N}{2}$. Second, N is sufficiently large such that $N+1$ is equivalent to N and $\frac{N}{N-2}$ is equivalent to 1. Third, the product of $\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z}$ within V_ϕ can be taken at its expected value. First, we state the marginal distribution over feature importance ϕ_i where i is an arbitrary feature importance $i \in d$. This given as

$$\phi_i|\mathcal{Z}, Y \sim t_{\mathcal{V}=N}(\hat{\phi}_i, V_{\phi_{ii}}s^2) \tag{A.6}$$

where $V_\phi = (\mathcal{Z}^T\text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}$. Recall each $\mathcal{Z}_{ij}$ is given $\sim$ Bern(.5) we use the third assumption to write V_ϕ is $\frac{\bar{\pi}N}{2}+1$ for the on diagonal elements and $\frac{\bar{\pi}N}{4}$ for the off diagonal elements. We can see this is the case considering that each element in $\mathcal{Z}$ is a Bern(.5) draw. We drop the $1's$ due to the first assumption.

Let $k = \frac{\bar{\pi}N}{2}$. It follows directly from Sherman Morrison that the i-th and j-th entries of V_ϕ are given as

$$(V_\phi)_{ij} = \begin{cases} \frac{2}{k} - \frac{2}{k(N+1)} & i = j \\ -\frac{2}{k(N+1)} & iNeqj \end{cases} \qquad (V_\phi)_{ii} = \frac{4}{\bar{\pi}(N+1)} \tag{A.7}$$

We see that the diagonals are the same. Thus, we take the *PTG* estimate in terms of a single marginal ϕ_i. Substituting in the s^2 estimate s_S^2 and using the second assumption, we write the variance of marginal ϕ_i as

$$\operatorname{Var}(\phi_i) = \frac{4s_S^2}{\bar{\pi}(N+1)} \frac{N}{N-2} \tag{A.8}$$

$$= \frac{4s_S^2}{\bar{\pi} \times N} = \frac{4s_S^2}{\bar{\pi} \times \operatorname{Var}(\phi_i)} \tag{A.9}$$

Because feature importance uncertainty is in the form of a credible interval, we use the normal approximation of $\operatorname{Var}(\phi_i)$ and write

$$N = \frac{4s_S^2}{\bar{\pi} \times \left[\frac{W}{\Phi^{-1}(\alpha)}\right]^2} \tag{A.10}$$

where W is the desired width, α is the desired confidence level, and $\Phi^{-1}(\alpha)$ is the two-tailed inverse normal CDF. Finally, we subtract the initial S samples. $\square$

A.2.2 Proposition 5.1

Before providing a proof for proposition 5.1, we note to readers that the claims are related to well known results in bayesian inference (e.g. similar results are proved in Bishop [2006]). We provide the proofs here to lend formal clarity to the properties of our explanations.

Convergence of Var(ϕ) Recall the posterior distribution of ϕ given in equation 5.3. In equation A.7, we see the on and off-diagonal elements of V_ϕ are given as $\frac{4}{\bar{\pi}(N+1)}$ and $-\frac{4}{\bar{\pi}N(N+1)}$ respectively (here replacing S with N to stay consistent with equation 5.3). Because we have $N \to \infty$, these values define V_ϕ due to the law of large numbers. Thus, as $N \to \infty$, V_ϕ goes to the null matrix and so does the uncertainty over ϕ.

Consistency of $\hat{\phi}$ Recall the mean of ϕ, denoted $\hat{\phi}$ given in equation 5.4. To establish consistency, we must show that $\hat{\phi}$ converges in probability to the true $\hat{\phi}$ as $N \to \infty$. To avoid confusing true $\hat{\phi}$ with the distribution over ϕ, we denote the true $\hat{\phi}$ as ϕ^*. Thus, we must show $\hat{\phi} \to_p \phi^*$ as $N \to \infty$. We write

$$\hat{\phi} = (\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))Y \tag{A.11}$$

$$= (\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))(\mathcal{Z}\phi^* + \epsilon) \tag{A.12}$$

Considering mean of ϵ is 0 and using law of large numbers,

$$= (\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}\mathcal{Z}^T \text{diag}(\Pi_x(\mathcal{Z}))\mathcal{Z}\phi^* = \phi^* \tag{A.13}$$

Convergence of Var(ϵ) Assume we have $N \to \infty$ so $\hat{\phi}$ converges to ϕ^*. The uncertainty over the error term is given as the variance of the distribution in equation 5.6. The variance of this generalized student's t distribution is given as converges to s^2 for large N. Recalling its definition, s^2 reduces to the local error of the model as $N \to \infty$. which is equivalent to the squared bias of the local model.

Appendix B

TalkToModel

B.1 XAI Question Bank

Here, we provide parses in the TalkToModel for the prototypical questions given in the XAI question bank Liao et al. [2020]. Our grammar can parse 30/31 core, prototypical questions, excluding socio-technical questions, demonstrating the grammar's broad coverage. Note, that questions provided in the question bank vary in how they are phrased regarding whether additional coreference is necessary. For instance, the question bank includes both questions of the form "what do you predict for this?" versus "what do you predict for Q?"). For conciseness, we write each question in the form where no further coreference is necessary ("what do you predict for Q?"). For the case where additional coreference is necessary it is straightforward to use the `previous_filter` operation to resolve the coreference. These results demonstrates the TalkToModel grammar is well equipped to support XAI questions.

Table B.1: Prototypical questions from the XAI question bank, parses in the TalkToModel grammar, and explanations of the parse.

What kind of output does the system give?
`function()`
System overview describes model output.

What does the system output mean?
`function()`
System overview describes meaning of model outputs (e.g., predict if someone has diabetes).

What kind of data was the system trained on?
`data(training_data)`
Data overview provides summary of dataset.

What is the sample size of the training data?
`count(training_data)`
Count provides number of items in the training data.

What is the distribution of the training data with with a given feature?
`statistic(training_data, feature_name)`
Statistic summarizes feature distribution

How accurate are the predictions?
`score(test_data, accuracy)`
Scoring functionality gets accuracy on test set

How often does the system make mistakes?
`incorrect(test_data, accuracy)`
Shows how often the model makes incorrect predictions.

In what situations is the system likely to be incorrect?
`mistakes(test_data)`
Summarizes the common situations mistakes are made and ways the system is wrong.

What kind of mistakes is the system likely to make?
`mistakes(test_data)`
Summarizes the common situations mistakes are made and ways the system is wrong.

How does the system make predictions?
`explain(test_data, feature_importance)`
Provides an overview of features used for making predictions

What features does the system consider?
`topk(test_data, all)`
Shows the feature ranking to demonstrate which features the model uses.

What would the system predict if a given feature A changes to..?
`predict(change(filter(test_data, id, A), feature, value, set))`
Shows predictions under single feature change.

Is feature X used or not used for the predictions
`important(test_data, X)`
Determines whether feature X is important for the prediction.

How should a given feature A change for this instance to get a different prediction Q?
`statistic(cfe(filter(test_data, id, Q, =), 100), summary, A)`
Summarizes changes to feature that will flip prediction.

What is the systems overall logic?
`explain(test_data, feature_importance) interaction(test_data)`
Provides first order feature importances and second order interaction effects to explain overall system logic.

Table B.2: Prototypical questions from the XAI question bank, parses in the TalkToModel grammar, and explanations of the parse (continued).

What features of instance Q determine the system's prediction of it?
```
topk(filter(test_data, id, Q, =), all)
```
Shows most important features that determine prediction.

Why are instance A and B given the same prediction?
```
explain(or(filter(test_data, id, A, =), filter(test_data, id B, =))), feature_importance)
```
Explains predictions for both instances by summarizing shared most important features across instances.

Why is this instance not predicted to be Q?
```
explain(filter(text_data, id, A, =), feature_importance, class=Q)
```
Explains alternate class prediction.

Why are instance A and B given different predictions?
```
statistic(or(explain(filter(text_data, id, A, =), feature_importance),
    explain(filter(text_data, id, B, =), feature_importance)), range, all)
```
Explanations reason for different prediction by contrasting feature importances.

How should instance A change to get a different prediction Q?
```
cfe(filter(test_data, id, A, =), 10, Q)
```
Computes several counterfactual explanations to provide different ways to get alternate predictions.

What kind of algorithm is used?
```
model()
```
Describes the ML model.

Why is the instance given this prediction?
```
explain(filter(test_data, id, X, =), feature, feature_importance)
```
Explains prediction for given instance with feature importance.

What is the minimum change required for instance A to get a different prediction Q?
```
cfe(filter(test_data, id, A, =), 1, Q)
```
Computes single, minimal counterfactual.

What kind of instance is predicted outcome Q?
```
statistic(filter(test_data, Y, Q, =), summary, all)
```
Summarizes instances with certain predicted outcome.

What is the scope of change permitted for instance A to still get the same prediction?
```
statistic(cfe(filter(test_data, id, A, =), 100), min, all)
```
Summarizes minimal changes to flip prediction.

What is the range of values permitted to for a given feature for this prediction on A to stay the same?
```
statistic(cfe(filter(test_data, id, A, =), 100), min, feature)
```
Summarizes minimal changes to certain feature that will flip prediction.

What kind of instance gets the same prediction Q?
```
statistic(filter(test_data, Y, Q, =))
```
Summarizes instances with the same prediction Q.

What would the system predict if instance A changes to...?
```
predict(and(change(filter(test_data, id, A), feature, value, set),
    change(filter(test_data, id, A), feature, value, set)...))
```
Shows predictions under potentially many different changes to instance.

What would the system predict for [a different instance A]?
```
predict(filter(test_data, feature, A, =))
```
Shows predictions for different instances according to filtering criteria.

What does [a machine learning terminology] mean?
```
define(term)
```
Defines a term.