SANDIA REPORT
SAND2021-11358
Printed September 2021

Sandia National Laboratories

SAGE Intrusion Detection System: Sensitivity Analysis Guided Explainability for Machine Learning

Michael R. Smith, Erin C.S. Acquesta, Arlo Ames, Alycia N. Carey, Christopher R. Cuellar, Richard V. Field, Trevor Maxfield, Scott Mitchell, Blake Moss, Elizabeth Morris, Megan Nyre-Yu, Ahmad Rushdi, Mallory Stites, Charles Smutz, Xin Zhou

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@osti.gov
Online ordering:	http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.gov
Online order:	https://classic.ntis.gov/help/order-methods

ABSTRACT

This report details the results of a three-fold investigation of sensitivity analysis (SA) for machine learning (ML) explainaiblity (MLE): (1) the mathematical assessment of the fidelity of an explanation with respect to a learned ML model, (2) quantifying the trustworthiness of a prediction, and (3) the impact of MLE on the efficiency of end-users through multiple users studies. We focused on the cybersecurity domain as the data is inherently non-intuitive. As ML is being using in an increasing number of domains, including domains where being wrong can elicit high consequences, MLE has been proposed as a means of generating trust in a learned ML models by end users. However, little analysis has been performed to determine *if* the explanations accurately represent the target model and they themselves *should* be trusted beyond subjective inspection. Current state-of-the-art MLE techniques only provide a list of important features based on heuristic measures and/or make certain assumptions about the data and the model which are not representative of the real-world data and models. Further, most are designed *without* considering the usefulness by an end-user in a broader context. To address these issues, we present a notion of explanation fidelity based on Shapley values from cooperative game theory. We find that all of the investigated MLE explainability methods produce explanations that are incongruent with the ML model that is being explained. This is because they make critical assumptions about feature independence and linear feature interactions for computational reasons. We also find that in deployed, explanations are rarely used due to a variety of reason including that there are several other tools which are trusted more than the explanations and there is little incentive to use the explanations. In the cases when the explanations are used, we found that there is the danger that explanations persuade the end users to wrongly accept false positives and false negatives. However, ML model developers and maintainers find the explanations more useful to help ensure that the ML model does not have obvious biases. In light of these findings, we suggest a number of future directions including developing MLE methods that directly model non-linear model interactions and including design principles that take into account the usefulness of explanations to the end user. We also augment explanations with a set of trustworthiness measures that measure geometric aspects of the data to determine *if* the model output should be trusted.

CONTENTS

Summary 13

1. Introduction 17

1.1. Scope of Research . . . 19
1.1.1. SA and MLE . . . 19
1.1.2. Trust Measures . . . 21
1.1.3. User Studies . . . 21
1.2. Lessons Learned . . . 21

2. Impact of Correlations on Classification 23

2.1. Experimental Design . . . 25
2.1.1. Data . . . 25
2.1.2. Training and Testing . . . 29
2.2. Design of Experiments . . . 33
2.3. Results . . . 34
2.3.1. Random Forest . . . 34
2.3.2. Multilayer Perceptron . . . 38
2.3.3. Support Vector Machine . . . 39
2.3.4. K-Nearest Neighbor . . . 40
2.3.5. Logistic Regression . . . 41
2.3.6. Gaussian Naive Bayes . . . 42
2.4. Feature Selection Algorithms . . . 42
2.4.1. Relief . . . 44
2.4.2. ReliefF . . . 45
2.5. Conclusion . . . 47
2.5.1. Future Work . . . 47

3. Correlation Preservation Sampling 48

3.1. Resmapling Approaches . . . 49
3.1.1. Empirical Methods . . . 50
3.1.2. Analytical Methods . . . 50
3.1.3. Comparison of Methods . . . 51
3.2. Correlation Preservation Sampling . . . 51
3.2.1. CorrPS-KDE . . . 51
3.2.2. CorrPS-TRV . . . 52
3.3. Experiments . . . 53
3.4. Conclusion . . . 55

4. Limitations of Current Machine Learning Explainability Methods 57
4.1. Global Sensitivity Analysis Methods 58
4.1.1. Sobol' Indices 58
4.1.2. Shapley Values 59
4.2. Trusting Learned Models 59
4.2.1. Black Box Explanation Methods 60
4.2.2. What Constitutes the Fidelity of an Explanation? 61
4.3. Empirical Examination of Explanation Fidelity 62
4.3.1. Synthetic Data 62
4.3.2. Ensemble of experts comparison 63
4.4. Discussion 65
4.5. Conclusion 68

5. Classification Trustworthiness 69
5.1. Geometric Trustworthiness 69
5.1.1. Training Proximity Metric 70
5.1.2. Extrapolation Metric 71
5.1.3. Class Ambiguity Metric 72
5.1.4. Measuring the Trust of a Region 72
5.2. Identifying Important Features 73
5.3. Synthetic Data Verification of the Trust Metrics 73
5.3.1. Synthetic Data Experimental Observations 74
5.3.2. Examination of Dense *Rings* & *Pluto* 8D 80
5.4. Experimental Results on PDFrate 84
5.5. Comparison with Out-of-Distribution Methods 84
5.5.1. Maximum Softmax Probability 86
5.5.2. Outlier Exposure 86
5.5.3. DeepMCDD 86
5.6. Implementation 87
5.7. Results 87
5.8. Future Work 88
5.9. Conclusion 89

6. User Study Results 90
6.1. Sage Advice? The Impacts of Explanations for Machine Learning Models on Human Decision-Making in Spam Detection [1] 90
6.1.1. Method 93
6.1.2. Procedure 96
6.1.3. Results 96
6.1.4. Discussion 100
6.2. Lessons Learned from xAI Deployment in a Cybersecurity Operations Setting 102
6.2.1. Methods 104

[1]Mallory C. Stites, Megan Nyre-Yu, Blake Moss, Charles Smutz, and Michael R. Smith. Sage Advice? The Impacts of Explanations for Machine Learning Models on Human Decision-Making in Spam Detection. In *International Conference on Human-Computer Interaction*, pp. 269-284. Springer, Cham, 2021.

6.2.2. Results ... 106
6.2.3. Discussion ... 110
6.2.4. Conclusions ... 112
6.3. Explainability for Model Maintainers ... 112
6.3.1. Objective and Research Question ... 112
6.3.2. Method ... 112
6.3.3. Findings ... 113
6.3.4. Discussion ... 116

7. Conclusions **118**

References **119**

References **119**

Appendices **129**

A. Machine Learning Models **129**
A.1. Random Forest ... 129
A.1.1. Multilayer Perceptron ... 133
A.1.2. Support Vector Machine ... 136
A.1.3. K-Nearest Neighbor ... 139
A.1.4. Logistic Regression ... 145
A.1.5. Gaussian Naive Bayes ... 148

B. Closed-Form Analytical Solution to Shapley Values **151**
B.0.1. Value of $v|_{\mathbb{S}}$ for Linear Models and Independent Variables ... 151
B.0.2. Value of $v|_{\mathbb{S}}$ for Linear Models and Correlated Variables ... 151
B.0.3. Value of $v|_{\mathbb{S}}$ for Nonlinear Models and Independent Variables ... 152
B.0.4. Value of $v|_{\mathbb{S}}$ for Nonlinear Models and Correlated Variables ... 152

C. Details for Classification Trust **153**
C.0.1. Scaling and Measuring Distance ... 153
C.0.2. Is This Just Another Classifier? ... 153

D. Ensemble of Experts Baseline **155**
D.1. Ensemble of Experts Baseline ... 155
D.2. Ensemble of Experts Comparison ... 157

E. Interview Questions **159**

LIST OF FIGURES

Figure 2-1. Permuting a feature of correlated variables 24
Figure 2-2. Histograms of individual feature data distributions 25
Figure 2-3. Scatter plots of feature interactions 26
Figure 2-4. Visual description of correlation for a variety of C_{F_1,F_2}. 27
Figure 2-5. Histograms of individual feature data distributions for a shifted feature 28
Figure 2-6. Permutation feature importance charts for various C_{F_1,F_2}. 32
Figure 2-7. Violin plots of accuracy and F1 score for random forest. 35
Figure 2-8. Violin plots of training/test accuracy for random forest. 36
Figure 2-9. Comparison of data distribution for $C_{F_1,F_2} = 1.0$ and 0.8. 37
Figure 2-10. Violin plots of feature importance for random forest. 38
Figure 2-11. Violin plots of accuracy and F1 score for multilayer perceptron 39
Figure 2-12. Violin plots of accuracy and F1 score for support vector machine 40
Figure 2-13. Violin plots of accuracy and F1 score for k-nearest neighbor 41
Figure 2-14. Violin plots of accuracy and F1 score for logistic regression 42
Figure 2-15. Violin plots of accuracy and F1 score for naive Bayes 43

Figure 3-1. SAGE approach to explaining machine learning models. The black arrows represent current steps in MLE. The blue portions highlight the contributions of using SAGE. 49
Figure 3-2. The results of sampling on the correlated petal width and petal length features from the iris data set showing a) the original dataset, b) the resampled dataset by perturbing only the petal width feature (bootstrapping), and c) the sampled data following the method used by LIME. 51
Figure 3-3. The results of sampling on the correlated petal width and petal length features from the iris data set showing a) the CorrPS-KDE, b) the heat map of the PDF from CorrPS-KDE, and c) the sampled data from CorrPS-TRV. 52
Figure 3-4. a) Histogram for the eight features in the synthetic dataset. There is slight class separation in feature 4 (upper right) and features 1 and 2 are correlated (upper left). b) The change in classification accuracy when removing a feature and retraining. 54
Figure 3-5. A comparison of the FI values using bootstrapping and classification accuracy (Default), Bootstrapping and model confidence (Model Confidence), and CorrPS-KDE and model confidence (Model Conf and CorrPS). Model confidence and CorrPS produce results more consistent with the established ground truth. 55

Figure 4-1. The fidelity of model explanations per feature. Here, a value of 1 is perfect fidelity. Non-linearity in the model has the greatest negative impact on fidelity 64
Figure 4-2. Mapping of GSA processes to xAI and highlighting current holes. 65

Figure 4-3. The results of sampling on the correlated petal width and petal length features from the iris data set showing a) the original dataset, b) the resampled dataset by perturbing only the petal width feature (bootstrapping), and c) the randomly sampled data. 67

Figure 5-1. a) **Training Proximity Metric**. Blue and orange training points are from different classes; shaded regions denote their clusters. The green triangle is well within the blue neighborhood; if the classifier assigns it blue, it is highly trustworthy. The classification of the yellow diamond has less trust as it is near both clusters. The purple square is not near either cluster, so its classification is also not trustworthy. b) **Extrapolation Metric**. Points outside the region are extrapolated, are considered less trustworthy. A point inside the hull is an interpolation of multiple points, so its classification is regarded as more trustworthy, regardless of it being assigned blue or orange. (Best viewed in color) 70

Figure 5-2. Extrapolation Metric categorization accuracy: inside ($d \leq 0$) vs. outside ($d > 0$). Training points are randomly generated in a unit hypersphere, and test points in the enclosing unit hypercube. For each curve we use n 3D convex hulls. We achieve 95 percent accuracy for the 20 dimensional case, and near 100 percent accuracy for 60+ dimensions. 71

Figure 5-3. Random line search for approximating the distance to the decision boundary. Given test point P (blue), the closest training point of a different classification is TP (orange). A line search between TP and P locates DP, which is approximately on the decision boundary. Going in random directions from DP locates RP with a different class than P. A line search from RP locates DP′, which is closer to P than DP. This is repeated. In practice we find a point on the decision boundary that is reasonably close to P. 72

Figure 5-4. Synthetic closed-geometry training (red and blue) and test (black) sets. 2D drawing of nD data, to scale. 73

Figure 5-5. Synthetic Gaussian training (red and blue) and test (black) sets. 2D drawing of nD data, to scale. The solid circle indicates a radius of 1σ. 74

Figure 5-6. t-SNE projection of the PDFrate data. a) **Malicious vs benign**. The 0 label represents the benign class while 1 represents malicious. It can be seen that there is some class separation. b) **PDFrate vs Evade**. The 0 label represents the evasion attack points and the 1 label represents the original PDFrate data. Visually, it is easy to see that the evasion attacks occur in less populated regions of the input space. 85

Figure 6-1. Example stimuli used in experiment. The left panel is an example of the 3 feature table condition; the right panel is an example of the 7 feature graph condition. The “email header info” and “email content info” were displayed on every trial, as was a model prediction. 95

Figure 6-2. Mean accuracy for each of the model percent accuracy conditions, collapsing across visual-ization condition. Accuracy was significantly higher in the 88% than 50% model accuracy condition. 97

Figure 6-3. Mean RT for each visualization condition, collapsing across model percent accuracy. RTs were significantly longer in the 7 feature table condition than baseline. 98
Figure 6-4. Mean accuracy by trial type and model percent accuracy, collapsing across visualization condition. Accuracy for false alarm trials was significantly higher in the 88% than 50% model accuracy condition; no other pairwise comparisons reached significance. 99
Figure 6-5. Mean accuracy for Miss trials only, by visualization condition and model percent accura-cy. In the 88% model accuracy condition, accuracy was significantly higher in baseline than all other conditions (except 3-features-graph). 99
Figure 6-6. Mean RTs for trials in which people did and did not comply with the model's prediction by model trust median split. 101
Figure 6-7. Obscured representation of the xAI tool output when expanded by analyst. . . . 104
Figure 6-8. Current explainability visualization with benign and malicious feature weights does not allow the user to manipulate the outputs in any way. Giving the analyst some control over the analysis interface will help them filter to information they think is relevant or better understand the context of the outputs for more effective decision making and perception of usefulness. 115
Figure 6-9. Explainability visualization shows "pdf_text_keyword_here" as a feature contributing to "malicious". However, it is unclear whether the high value is due to a large number of the word "here" in the pdf text, or due to a small number of the word "here" in the pdf text. 116

Figure A-1. Example of splits for F_1 and F_2 in a decision tree. 130
Figure A-2. Example split for F_1 and F_2 in a decision tree. 131
Figure A-3. Example output from MLP over F_1, F_2. 134
Figure A-4. Example of MLP weights. 136
Figure A-5. SVM decision boundary over F_1 and F_2. 137
Figure A-6. SVM decision boundary over F_1 and F_8. 137
Figure A-7. SVM Results for various shift sizes. 138
Figure A-8. Proportion of correlated points in K-nearest neighbors. 141
Figure A-9. Average proportion of correlated points in K-nearest neighbors. 141
Figure A-10. Average proportion of correlated points in ε-neighborhood. 142
Figure A-11. Proportion of correlated points in ε-neighborhoods. 143
Figure A-12. Proportion of correlated points in ε-neighborhoods. 144
Figure A-13. Average proportion of correlated points in K-nearest neighbors for various C_{F_1,F_2} 144
Figure A-14. Average proportion of correlated points in K-nearest neighbors, $C_{F_1,F_2} = 0.0$. . . 145
Figure A-15. Logistic regression decision boundary over F_1 and F_2. 146
Figure A-16. Logistic regression decision boundary over F_1 and F_8. 147
Figure A-17. Values for $P(x_8|y=1)$ and $P(x_8|y=2)$ in the shifted data. 150

Figure E-1. Interface of the explanation integrated into the standard user interface used the incident responders. 160

LIST OF TABLES

Table 2-1. Distribution of F_8 for $\mu = -0.25$. 34

Table 3-1. Accuracies from the examined learning algorithms on the synthetic datasets. . . . 55

Table 4-1. Examples explanations on prediction of malicious PDFs by an ensemble of experts and SHAP-Dep and TreeSHAP. 66

Table 5-1. Training data locations. 74
Table 5-2. Training data sizes. 75
Table 5-3. Test data location. *Far* outside the training data but closer to one set, far from the other. *In* is closer to that other set in *Rings*. For the other sets, *In* points are a subset of one class's distribution. *Mid* is between the two classes' distributions. A is from the distribution of the first class, and B is from the distribution of the second class. For planes, the *In* test data is already in-distribution, and the classes are symmetric, so no A and B distribution test sets are needed. For Normals, the *Far*, *In* and *Mid* test sets are $(D-1)$ hyperballs in the hyperplane with the given last coordinate. For Normals, the Indistribution test sets are Gaussians with the same offset and sigma as the training data. Each test data set has 100 points. 75
Table 5-4. Experimental observations on the *Planes* data 77
Table 5-5. Experimental observations on the *Rings* and *Pluto* datasets. The average Proximity values include outliers as zeros. 79
Table 5-6. *Rings* & *Pluto* 8D **dense** experimental data. 81
Table 5-7. 3D Normals experimental data. Here we report the Proximity strength averages *excluding* the outliers. For the SVM and RF classifiers, we report the number of test points assigned to each of class A and B. 82
Table 5-8. 8D Normals experimental data. Here we report the Proximity strength averages *excluding* the outliers. 83
Table 5-9. Summary of the trust metrics on the PDFrate data set. The average Training Proximity Metric is the strength values returned by HDBscan where 1 indicates a strong relationship with a cluster and 0 represents an outlier. The Extrapolation Metric indicates the distance from a convex hull around the training data. A value of 0 indicates that the test point is on the convex hull. The Class Ambiguity Metric indicates the distances from the classification boundary. 86
Table 5-10. Contagio Dataset Results. TP: correctly predicted ood at recall=0.95 88

Table 6-1. Feature counts across the stimulus set. Features marked with * indicate that the feature was as a "stimulus information feature," and was presented on each trial regardless of experimental condition. 94

Table 6-2. Trust in Automation Scale 95
Table 6-3. Mean accuracy and response times by visualization type and model accuracy condition. 97
Table 6-4. Likelihood of compliance with model's prediction by model trust score and overall model accuracy. 100
Table 6-5. Data collected from 17 SCOT entries. LB=LaikaBoss 107
Table 6-6. Time to close events where explanations were viewed. 107
Table 6-7. Analyst agreement with classifier in events where explanations were viewed. . . . 108
Table 6-8. Unique analysts with SCOT entries for events where explanations were viewed. 108
Table 6-9. Number of total SCOT views for events where explanations were viewed 108
Table 6-10. Survey data from one analyst (of 11 possible, response rate = 9%), $n = 1$. 109
Table 6-11. Practical considerations for xAI deployment 111

Table A-1. Counts of instances where a given feature was used in the parent node for a decision tree (out of 10,000), for 8 features per node. 132
Table A-2. Counts of instances where a given feature was used in the parent node for a decision tree (out of 10,000), for 2 features per node. 133
Table A-3. Example MLP weights. 135
Table A-4. Approximation to MLP in terms of F_1 and F_2. 135
Table A-5. Mean and standard deviation of SVM test-set accuracies for various shift sizes. . 139

Table D-1. Counts of analyst features by how well they are represented by the machine learned feature set and how many of the machine leared features are required to provide coverage 157

SUMMARY

This report summarizes the research and results of the two year LDRD: *SAGE Intrusion Detection System: Sensitivity Analysis Guided Explainability for Machine Learning*. The intention of SAGE was to examine machine learning explainability (MLE) with respect to the observation that most MLE methods are based on heuristics and lack verifiable assessment of the value they provide to the end user. In particular, SAGE intended to: (1) ground MLE in the mathematical principles of uncertainty quantification and sensitivity analysis, (2) augment explanations with trust measures based on the geometric relationship of a test point with the training data, and (3) integrate MLE into an existing workflows for the purpose of examining the impact of MLE in real-world settings and allowing ML practitioners to evaluate. We, therefore, had a parallel paths of investigation focusing on the comparison of perturbation-based and game-theoretic approaches to black-box MLE methods.

Sensitivity analysis (SA) was chosen because it is similar to many of the perturbation-based MLE methods that lack a principled approach to provide theoretical guarantees and uncertainty bounds. Upon close investigation of the MLE methods it was identified that they make strong assumptions to avoid computational overhead: (1) feature independence, and (2) linear feature interactions within the model. The independence assumption provides the convenience of avoiding the need to compute the full joint probability distribution over the full feature set and but results in the sampling of non-realizable inputs that are out-of-distribution from the training data. It is worth noting that this assumption spans all applications for which sampling methods are utilized and the consequences of misusing the independence assumption will apply to all application contexts for which the independence assumption is simply not true. While the first assumption has critical consequences, we found that the model linearity assumption has a larger impact on the fidelity of the explanations to the ML model. Faithfully capturing non-linear interactions is a challenging problem that needs further study.

To increase trust, MLE techniques provide a set of features that influence the output of an ML model. However, they do not provide a measure of how much the output should be trusted—that is left to the user to decide. Current practices use the confidences from the output of a model, however, model confidences have been shown to be uninformative and often require very large perturbations to cause noticeable changes. We, therefore, examine geometric relationship between a test point and the training data. We explicitly seek to measure three salient properties: (1) is the test point in a dense region of the training space, (2) is the test point an extrapolation or an interpolation between training points, and (3) is the test point near the classification boundary? Preliminary results suggest that these properties may be able to detect out-of-distribution data and adversarial attacks.

We worked with Enterprise security at Sandia National Labs to investigate how MLE impacts an end-user in real-world situations. We chose cyber-security for maximum relevance and benefit to

real-world practices. First, cyber data are inherently non-intuitive—as opposed to images and text—so we hypothesized that explanations would provide more benefit, e.g., in helping to triage cyber attacks. Second, we were able to measure impact directly without any major changes to cyber analysts' current workflow. Despite a desire to have explanations, we found that explanations were often not used or consulted. In some cases, we found that explanations persuaded the end-user to accept a false negative or false positive rather than explain the decision process.

While our research highlights several deficiencies in MLE, we conclude and are optimistic that MLE *could* have a larger impact if we framed MLE within the context of model credibility, similar to simulation models used in other high-consequence applications. A laboratory such as Sandia National Laboratories, which has a rich history in verifying and validating models relating to many national security related domains provides fertile ground for developing such processes. Our work can be leveraged within the budding scientific ML as well as classical ML paradigms. For usability, we advocate that the MLE system be designed with the end-user in mind and ensuring that explanations are useful to the end-user. Current processes often build MLE in isolation of the application space and other context specific constraints.

Our research also highlights that MLE is inherently a multi-disciplinary effort. The SAGE team comprised ML and MLE expertise, cybersecurity practitioners who also developed ML models for the cyber workflow, mathematicians, and cognitive scientists. Additionally human-computer interaction specialists could have also been employed to help display the explanations. Future development of MLE techniques should take into consideration these perspectives.

NOMENCLATURE

Table 0-1.

Abbreviation	Definition
DOE	Department of Energy
DoE	Design of Experiments
FI	Feature Importance
GUI	Graphical User Interface
ID	In Distribution
ML	Machine Learning
MLE	Machine Learning Explainability
OOD	Out-of-distribution
QoI	Quantity of Interest
SA	Sensitivity Analysis
SAGE	Sensitivity Analysis Guided Explainability
UQ	Uncertainty Quantification
V&V	Validation and Verification
XAI	explainable artificial intelligence

1. INTRODUCTION

Machine learning (ML) algorithms are utilized in an increasing number of high-consequence and sensitive application areas; examples include malware detection [103], autonomous vehicles [29], and medical diagnoses [28]. This increase in use and ubiquity of ML in high-consequence applications highlights the need to understand how an ML model behaves along with why a particular output is given to ensure safety, preserve fairness, and reduce biases as much as possible such that the model is trusted. While many of these application areas are intended to work autonomously, there is also a need to explain decisions to a decision maker with a human-in-the-loop. Both scenarios provide unique constraints in validating the model before deployment, ensuring their correctness, and how to explain complex decision processes within a learned ML model. In response, ML explainability (MLE) and explainable artificial intelligence (XAI) have emerged as a budding research field [4]. However, the MLE and XAI methods inherently lack rigor and formality congruent with the state-of-the-art in model credibility assessments for computational simulation and engineering (CS&E) models. To explain this disconnect one should be reminded that the standard use cases for which ML models are employed are specific to the application context for which we lack known properties or equations as is done in computational simulation. The ML model is expected to learn the properties. It is also worth noting that core to a formal model credibility assessment, and sensitivity analysis specifically, is the proper treatment of uncertainty quantification (UQ). Today, UQ for ML is in the beginnings of development [113] and are not explored for MLE. Developing the underlying mathematical principles is essential to allow for the proper treatment of UQ in MLE.

In this LDRD, *SAGE Intrusion Detection System: Sensitivity Analysis Guided Explainability for Machine Learning*, we consider the challenges of adapting the principles from design of experiments for sensitivity analysis (SA) to define a mathematical framework for MLE methods (what we term *SA Guided Explainability* or SAGE); including the downstream effects of explanations on the end-user. We implement MLE methods into the actual workflow of Enterprise Security at Sandia National Labs and measure the impact of explanations in terms of decreased response time and increased accuracy in detecting attacks. SAGE was motivated by the observations that most black-box MLE research was: (1) similar to sensitivity analysis although more heuristic and lacked mathematical principles—not setting up the correct pillars for UQ, (2) success was declared by the ML developer and user studies were limited or non-existent, and (3) the emphasis of being able to trust ML algorithms in high-consequence applications. Therefore, SAGE sought to bridge several expertise areas across Sandia National Laboratories, namely, sensitivity analysis (SA) used in validation and verification (V&V), ML, and cognitive psychology to develop principled approaches to MLE in a high-consequence application—cybersecurity.

The format of this report follows a collection of reports on specific research questions that were submitted or prepared for internal and external communication. Thus, each chapter could stand alone and there is some repeated information. This introduction serves as a high-level conglomeration of main take points and lessons learned. We will conclude the report with future directions. As this LDRD is highly diverse, we will provide a brief overview of what we mean by certain terms in the following list and refer the reader to existing background material for additional information:

- **Sensitivity Analysis.** *Sensitivity analysis* (SA) is the mathematical study of how uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input. There a multiple types of SA including global and local. See Saltelli et al. [98] for more background. While we primarily leverage examine Sobol' indices, we also examine the recent finding of the relationship between Shapley values [104] from cooperative game theory and Sobol' indices [83].

- **Machine Learning.** The field of *Machine Learning* (ML) comprises the study and development of algorithms that are able to learn and adapt to training data without being explicitly instructed to do so. Common examples are learning to classify, clustering objects, and regression. There are several books that can referenced [35, 33, 10].

- **Machine Learning Explainability.** *Machine learning explainability* (MLE) is a subfield of ML that examines how to explain the decision process made by a learned model *and* convey that information to an end-user such that it is understandable. There are several surveys that can be consulted [1, 20, 117, 76].

- **Adversarial and Out-of-Distribution Detection**. Adversarial attacks and concept drift are two common concepts in ML where the data changes intentionally or unintentionally over time and cause an ML model to have low confidence. These are both budding research fields and related to MLE in our desire to augment explanations with a trust measure. For an overview of detecting these types of points in deep learning see the survey by Bulusu et al. [15].

- **Evaluation of Explanations.** While most explainability methods do have some type of evaluation, there is no established method of how to compare them. Warnekcke et al. [122] evaluated the explanations for security application based on measurable attributes that are assumed to correlate with usefulness to an end-user. A more end-user centric study was conducted by Miller [74] using insights from the social sciences in which he provides many perspectives and findings in how humans explain concepts to each other—many of which are lacking in MLE. Most notably, the fact that an explanation is often a conversation which is inherently lacking in the static explanations provided by MLE methods. A glaring hole is the fact that there is no defined and accepted definition of what constitutes a good explanation.

This list is not intended to be exhaustive and omits work dealing with specific use cases, such as intrusion detection and the specific nuances to that application, as well as human-computer interfaces and many more related topics. This all highlights the difficulty in effectively developing and deploying MLE methods.

1.1. Scope of Research

1.1.1. SA and MLE

Fundamental to our research efforts is the relationship between MLE and SA. As these are two large fields of study, we establish the scope with respect to each field that our team explored. Emphasizing the importance to establish consistent terminology and examine the objectives and assumptions in each field, this section will establish the preliminary notion while providing a high-level summary of their relationship and the challenges faced when applying SA for MLE.

At the foundation of our exploration into SA for MLE we focus on supervised machine learned models. In supervised ML, the goal is to learn a mapping f from a set of n input-output pairs $d = \{x_i, y_i\}_{i=1}^{n}$, commonly referred to as the training data. The machine learned model, $f : X \to Y$ maps input vectors $X = \{x_i\}_{i=1}^{n}$ to their corresponding output vectors $Y = \{y_i\}_{i=1}^{n}$ so that f maximizes the likelihood of the relationship between X and Y. The ML algorithm, $\mathscr{F}$, used to learn f will also introduce an additional collection of m hyper-parameters, $\Theta = \{\theta_i\}_{i=1}^{m}$. The selection of the ML algorithm (e.g., random forests, support vector machine, neural networks) and its architecture will predefine the space of hyperparameters. Once the algorithm has been determined, the objective of the learning algorithm can further be categorized as either *regression* or *classification*. We will address both regression and classification methods in this report; noting that regression algorithms lend themselves more readily to SA, while classification methods introduce more of our interesting challenges. To further determine the suitability of SA for MLE we will need to first outline the basic definitions and assumptions of each.

We define MLE as a set of techniques for explaining the inner-workings of a learned model. As many models are sufficiently complex, so much so that presenting the inner-working of a model is not decipherable by an end-user, many explanation methods for an ML model present a ranked set of *important features* that contribute the most to the model prediction. While there are many approaches to explainability, identifying important features is the most commonly used. Since determining what features are important depends on the definition of importance, we define *feature importance* (FI) as weighting or ranking of the input features based on how essential each feature is for providing the given prediction. For example, if that feature is removed (or perturbed) and we would get the same prediction we say that this feature has low FI. Alternatively, if a feature is removed (or perturbed) and the model provides a different prediction, we consider this feature to have high FI. Critical to these failures of FI-based MLE will be explained in more detail in Section 4. For now we summarize that most of the approaches assume that input features are independent and forgo the inference needed to capture any higher-order (order > 2) interactions in the model.

Most SA methods will also have these same limiting assumptions with regards to independence and higher-order interactions in the model. Although there are more recent advancements in SA methods that account for correlated inputs and there are theoretical guarantees regarding the higher-order interactions, all of these methods are limited in practice by their computational complexity [42]. The value that SA can still provide, that MLE lacks, is the mathematical foundation and theory for which the numerical implementations are derived. The goal of a SA is to verify the consistency of the model behavior or to assess the robustness of simulation results to uncertain inputs or model assumptions. SA can be defined as “the study of how the uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input" [98]. Critical to SA is proper treatment of Uncertainty Quantification (UQ), which fundamentally requires formal mathematical and statistical considerations for representing random variables and their propagation through a model to provide an appropriate estimate of the resulting output uncertainties. The extension to SA, for which we determine what sources of uncertainty in the input of the model impact the resulting output uncertainty the most is best cast into a Design of Experiments (DoE) framework. DoE provides the assessment of a given causation hypothesis so that the factors of influence are well-defined, sufficient replicates are run to empirically represent the random output while simulating uncontrolled sources of additional random behaviors, and that the target Quantity of Interest (QoI) is also well-defined and measurable. In this context, we identify that SA will determine a ranked list of important sources of uncertainty. When we consider the SA for MLE, we note that the sources of uncertainty most predominately come from our feature space; implying that SA provides inference regarding the important features. But what does *important* mean? Since SA will apportion the uncertainty of the input features as a ranking of their influence on the uncertainty of the prediction, the result is a measure of importance that more closely relates to class confusion. It is important to note that this is distinct from the standard goals of MLE, although not entirely useless. Heuristically both SA and MLE have similar methodology, especially when we compare perturbation-based MLE with variance-based SA. Therefore, the comparison between the two is still worth consideration.

As we noted earlier, the opportunity to apply SA to an ML model is better suited when the objective of the ML algorithm is to learn a regression model. For this reason, we will provide an intuitive regression example so that we can ground the difference between MLE and SA. Consider the following linear regression equation:

$$y = 10x_1 + 0.1x_2 + 0.0001x_3 \tag{1.1}$$

The goal of most FI-based MLE methods is to determine each features $\{x_1, x_2, x_3\}$ importance in the outcome of y. Assuming a normalized feature space, this can easily be inferred by examining the weights of the model. Note, a small change is x_3 will only result is an infinitesimal change in y. Alternatively, even a small change in x_1 can result in a substantial change in y. This is because the weight of x_1 is four orders of magnitude greater than x_3.

Now consider what happens we we apply SA to equation 1.1. In this context, we would assume that each one of the features can be modeled as a continuous random variable. Assuming each feature is normally distributed with zero mean, $x_i \sim \mathcal{N}(0, \sigma_i^2)$. The results of an SA analysis will be consistent with the result of an FI-based MLE, if and only if, the features are independent and identically distributed (i.e., $\sigma_1^2 = \sigma_2^2 = \sigma_3^2$). Instead, if σ_3^2 is more than four orders of magnitude greater than σ_1^2, the results of SA will rank x_3 greater than x_1 because the uncertainty inherent to x_3 is sufficiently larger than x_1.

The reason we focus on regression to start is because SA requires a continuous random variable for the QoI. Alternatively, if we consider classification instead, the outcome of the ML model is defined on the space of discrete realizations. In this case, continuous output can easily be provided using heuristics such as the distance of a point from the hyperplane in a support vector machine or the purity of a leaf node in a decision tree. Some methods, such as Platt’s scaling [90] go even further to make the output more like a probability by using a non-linear transformation. Both, the choice of the QoI for classification and the manner for translating the discrete class prediction to a continuous value have ramifications on the results of the SA.

1.1.2. Trust Measures

The notion of adversarial robustness and out-of-distribution (OOD) detection have been explored mostly in isolation, but examine the task determining if the prediction from an ML model for data point should be trusted. The goal is to identify data points that are different from the training set either due to naturally evolving data distribution or intentional modification. We do not attempt to focus exclusively on OOD or adversarially perturbed data points. Our focus is on quantifying the relationship of the test point to the training point geometrically. We are unique in the sense that we do not explicitly try to use an ML model to quantify trust (Chapter 5).

1.1.3. User Studies

Examining user interaction with automated and ML systems is a large research field encompassing various research questions of its own. In SAGE, we focused on quantitatively measuring the effectiveness of MLE. While there are many different directions that could have been pursued, we considered multiple user studies to (1) determine *if* explanations are useful, (2) what methods for displaying explanations are the most effective, and (3) if explanations make a significant impact in real-world deployments. Ultimately, we implemented MLE as part of the Enterprise Security workflow and measured how much explanations affect the performance of the cybersecurity experts. To fill gaps in understanding perceived usefulness and usability of the explanations in the Enterprise Security context, we also conducted a qualitative study with experts in different roles that interface with the ML model. The results revealed insights around how ML explanations might be used (or not) as well as information requirements for iterating design of the explainability tool. Results are presented in Chapter 6.

1.2. Lessons Learned

Through our research as part of SAGE, we found that while MLE is sorely needed, the current state-of-the-art techniques are severely lacking in terms of providing high fidelity to the model that is being explained and in terms of their usefulness to the end user. Our intention was to develop new methods that integrated principles from SA and to significantly improve the effectiveness of the end users. However, there are several open research areas that need to be addressed to close the gap between SA and MLE in addition to making the explanations useful to the end user.

The salient lessons learned can be summarized as:

- Despite features dependence is often ignored, correlated features can play a significant role in classification (Chapter 2).

- We proposed a method for sampling that preserves correlations between features (Chapter 3). We independently discovered this as Aas et al. [1], however, we present our derivation here as we also compare with some kernel-density estimators and we examine its effects in higher-dimensions (Chaper 4). One of the key limiting factor is that modelling correlations requires significantly more samples to properly model the interactions in higher dimensions. In cases, where this is not met, the estimations are not as beneficial. Aas et al. only considered up to 10 features, and, thus, did not discover this limitation.
- Mathematically, explanations have low fidelity for representing the ML models that they are explaining (Chapter 4). This is due to several assumptions that MLE techniques make, including:
  - **Feature Independence.** To minimize the computational expense, features are assumed to be independent. This assumption, of course, does not hold in real-world datasets and can have detrimental effects as sampled data represents out-of-distribution data and data that is not physically realizable.
  - **Linear Feature Interaction.** ML models are desired to model complex feature interactions that interact in non-linear ways. However, it is unclear how to identify these interactions by MLE algorithm and how to present them if they were found.
  - **Model Output as QoI.** One challenge for MLE is choosing a suitable QoI. Most models provide a confidence measure, however, many of these have been shown to be mostly meaningless as the model is overly confident in most of the input space [44].
- We observed that many analysts use the outputs of the ML model, but do not utilize the explanations from an MLE method. In the worst case, as observed in a study of non-analysts (Chapter 6), a user can be persuaded to accept a false-negative or a false-positive.
- Geometric trust metrics show promise in identifying various types of non-training data (Chapter 5). However, future work needs to address methods to speed up computation in order to scale to larger input sizes.
- Explanations are more beneficial to an analyst that needs to understand the model's underlying logic in order to make a decision about an observation. This could include a subset of incident handling scenarios. However, more often MLE is of primary benefit to ML model developers and maintainers (Chapter 6).
- The current form for providing explanations can help in some cases to draw an analyst's attention to a particular (or subset of) feature(s) of an observation. However, the static representation does not allow a user to understand the model itself. In our user-study interviews, an interactive model probing would be the most beneficial and aligns with the finds of Tim Miller [74] that an explanation is a 2-way conversation in exchanging information.

2. IMPACT OF CORRELATIONS ON CLASSIFICATION

It is commonly acknowledged that strong correlations between features provide redundant information to machine learning models (see [57], [40], [127], and [65] to name a few). These ideas have persisted since at least 1964, where in [34] the idea of the *merit* of a subset of features was proposed:

$$\text{Merit}_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

Where s is a feature subset of k features, and $\overline{r_{cf}}$ and $\overline{r_{ff}}$ indicate the average feature-class correlation and feature-feature correlation, respectively. This balances relevancy and redundancy of a feature subset. Strong feature-class correlations (high relevancy) increase the merit of a subset, while strong feature-feature correlations (high redundancy) decrease the merit.

Correlation feature selection is based on this idea of merit [41], defining the optimal subset of k features as

$$CFS = \max_{S_k} \left[\frac{\sum_{i=1}^{k} r_{cf_i}}{\sqrt{k + 2(\sum_{i=1,j=1,i\neq j}^{k} r_{f_i,f_j})}} \right]$$

Again, maximizing relevance while minimizing redundancy.

Work in [127] defines a method of feature selection utilizing similar ideas of relevancy and redundancy. The authors propose a relevance analysis and then a redundancy analysis of the relevant subset. Their relevance analysis involves the calculation of feature-class correlation for each feature to select a relevant subset. At that point, rather than computing nearly N^2 feature-feature correlations, an approximate Markov blanket is used to determine an optimal feature subset.

Two methods, Relief [55], and ReliefF [58] are implemented and analyzed on an experimental dataset used throughout this chapter described in Section 2.4. Relief was originally proposed as a method that selects features while being sensitive to feature interaction. This is done by noting the distance between feature values of nearby points of the same and different target labels, a different approach than correlation computation.

All of the above listed methods of feature selection are considered **filter methods**, where a subset of features is selected before the learning process is even started. Thus, they are applicable to nearly any learning scenario. Two other methods of feature selection include **wrapper methods**, where feature subset selection is included in a validation process, and **embedded methods**, where the model chooses the feature subset as a part of training (i.e. lasso regression).

Even though some feature selection algorithms are sensitive to correlations and other feature interactions, this long-standing bias against correlated features has lead to a relatively standard assumption in machine learning and data science: that our features should remain relatively uncorrelated. We argue that not only does this remove information from the model, this is especially problematic from the perspective of machine learning explainability.

In permutation feature importance [32], for example, a feature is randomly permuted among the data points, as to break any relationship between that feature and the target output. This permuted data is used in a model and any decrease in accuracy is attributed as the "importance" of that feature. However, as seen in Figure 2-1, this can cause a significant number of permuted points to lie outside the training distribution of the data. These are regions where it is difficult, if not impossible, to make any guarantees of the model's performance. In [50] the author discusses the extrapolative nature of these out of bag sample points and how they lead to misleading feature importance measures.

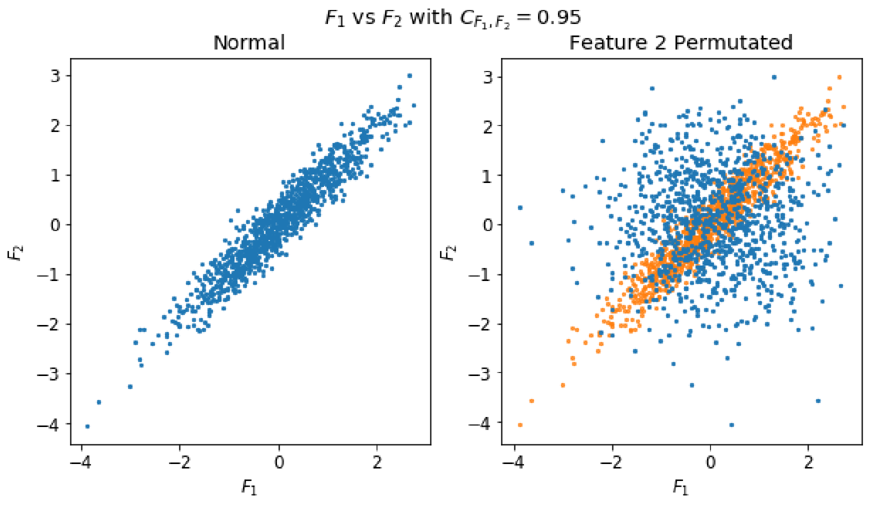


Figure 2-1. Permutation of a feature belonging to a correlated pair ($c = 0.95$).

Similar issues arise in many other explainability methods, such as partial dependence plots (PDP), Shapley additive explanations (SHAP), and local interpretable model-agnostic explanations (LIME) [76] (See Chapter 3).

From several angles we see that correlations can be difficult for ML practitioners. They diminish the abilities of black-box explainers, add complexity to feature selection, and increase the computational cost of learning (although this is increasingly less of an issue, thanks to Moore's law). However, *we hypothesize that correlated variables provide discriminative power for classification when using certain machine learning algorithms.* Through rigorous design of experiments (DoE) we propose a situation in which correlation exists as the single defining means of separability between two classes, and see that several machine learning algorithms are able to pick up on this. Thus, we advocate for a better understanding of the effects of interacting features. We examine correlations in feature selection algorithms and ML classification algorithms. We first describe the synthetic data and our experimental set up. We then present results on feature selection algorithms and classification algorithms.

2.1. Experimental Design

2.1.1. Data

To test the given hypothesis, a synthetic data set was designed in order to isolate the effects of correlation on a machine learning model's ability to classify data. This data consists of eight features, F_i for $i = 1, 2, \ldots 8$, which are initially all sampled from a normal distribution centered around 0 with a standard deviation of 1. Thus as the number of samples increases, the two-sample Kolmogorov-Smirnov test will converge towards zero for *all* combinations of features.

To utilize correlation as a means of distinguishing between the two classes, correlation is applied to one class. Features F_1 and F_2 are correlated at various intensities, measured through the Pearson correlation coefficient C_{F_1,F_2} of the two features. Note that when $C_{F_1,F_2} = 1$ we have that $F_1 = F_2$ for every instance in the correlated class. However, when analyzed individually, each feature still originates from the same distribution, even with complete correlation. See Figure 2-2.

Analyzed pairwise, every combination of features except F_1, F_2 will look uncorrelated. See Figure 2-3 on the next page.

The idea of all of this is that when analyzed from the perspective of any particular feature the data looks identical for both classes. However when a higher-order interaction is taken into account, the data becomes separable.

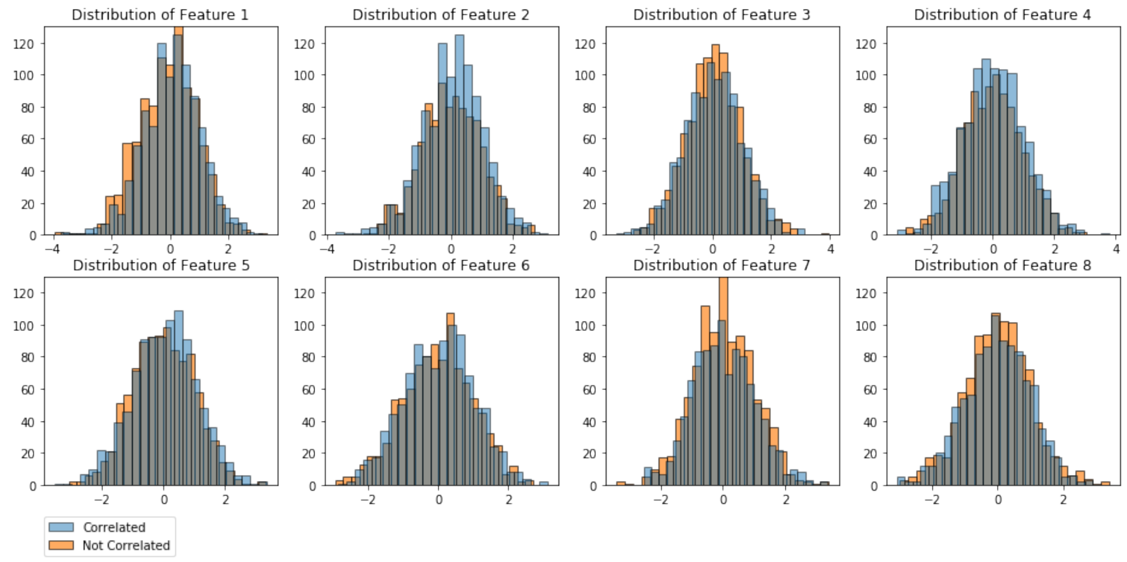


Figure 2-2. Histograms of the distribution of data in each feature, for $C_{F_1,F_2} = 1$ for the correlated class. Note that the blue distributions for F_1 and F_2 are identical.

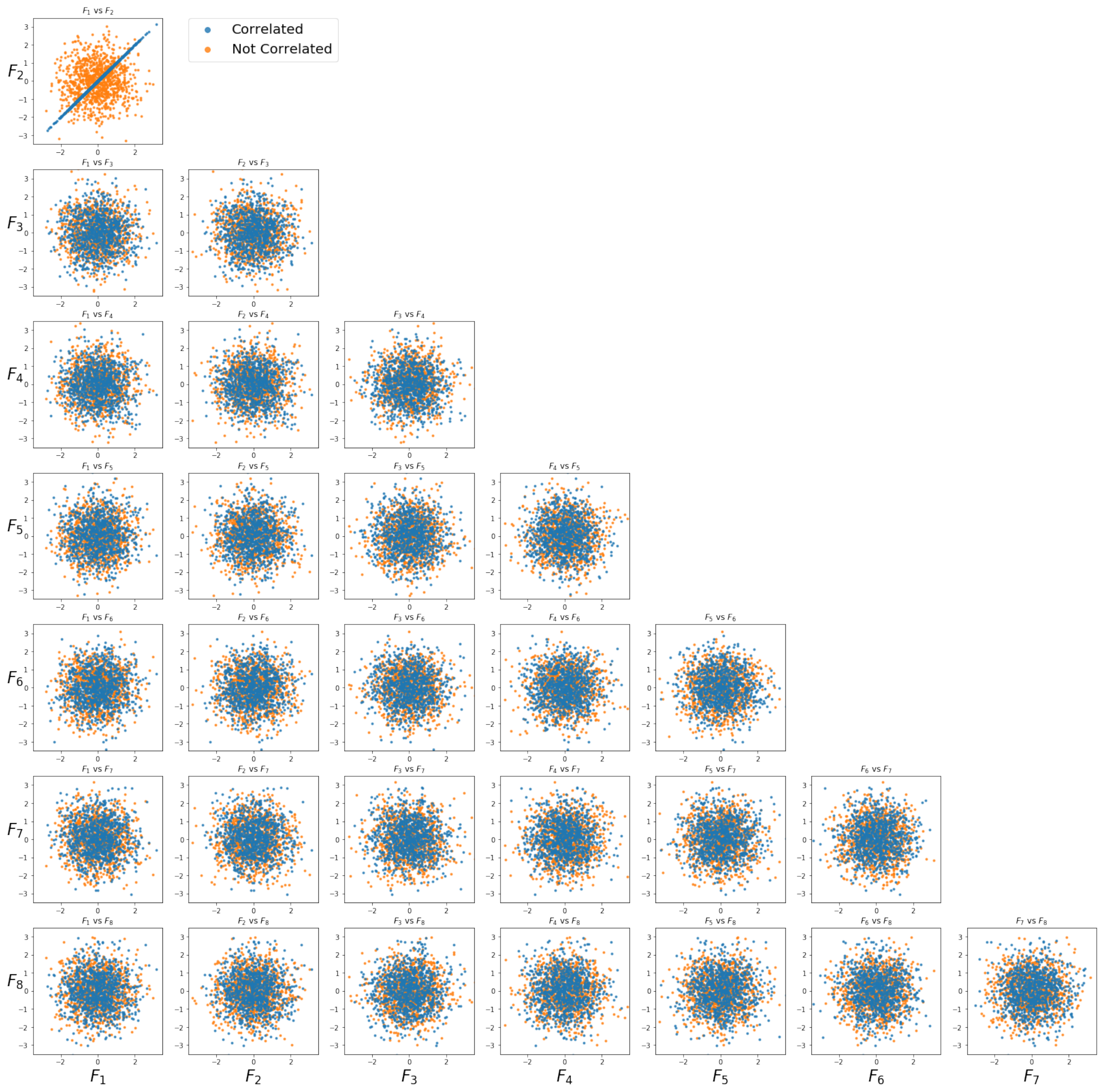
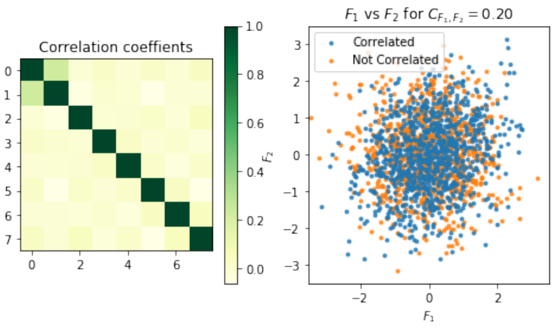
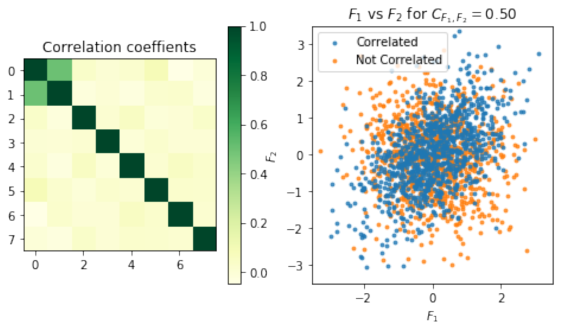


Figure 2-3. Scatter plot of each pair of features, for $C_{F_1,F_2} = 1$ for the correlated class.

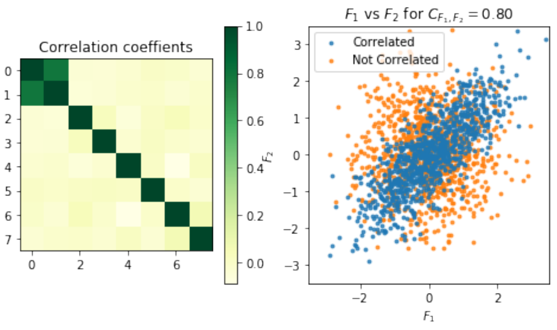
To visually describe the effects of correlation on two features, Figure 2-4 gives a scatter of F_1 vs F_2 for various correlation coefficients C_{F_1,F_2}. As well, a matrix of Pearson correlation coefficients between features is given to show how randomly generating identical distributions for each feature yields relatively little correlation amongst other feature pairs.



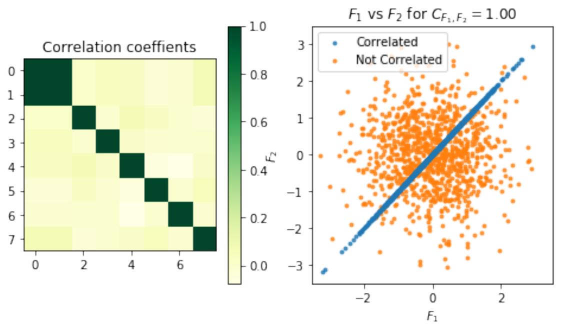
(a) Visual description of correlation, $C_{F_1,F_2} = 0.2$



(b) Visual description of correlation, $C_{F_1,F_2} = 0.5$



(c) Visual description of correlation, $C_{F_1,F_2} = 0.8$



(d) Visual description of correlation, $C_{F_1,F_2} = 1.0$

Figure 2-4. F_1 vs F_2 for a variety of C_{F_1,F_2} alongside the Pearson correlation coefficients for the data.

Note that a secondary experiment utilizes a feature with a shifted mean to give some notion of one-dimensional class separability. This is implemented as a shift to the mean of a feature (F_8) by a positive value for one class and a negative value for the other. Figure 2-5 on the next page gives the distributions of the values for each feature for $C_{F_1,F_2} = 1$ and a gap of 0.5 on the means of the two classes in F_8.

To summarize the data consists of two classes, one of which contains a correlation between F_1 and F_2. The other contains no intentional correlations. With respect to a single feature at a time, the classes have no separability. All features are sampled from a normal distribution with mean 0 and standard deviation of 1. Since there is absolutely no correlation between any single feature and class membership, every feature could be considered *irrelevant*, if we define irrelevancy without regard to higher-order interactions between features. As for *redundancy*, in the correlated class we could consider F_1 and F_2 to provide redundant information to the degree provided by their correlation coefficient. Even in the case where one of F_1 and F_2 is not dropped due to redundancy (it is apparent that there is information contained in the *lack* of correlation of

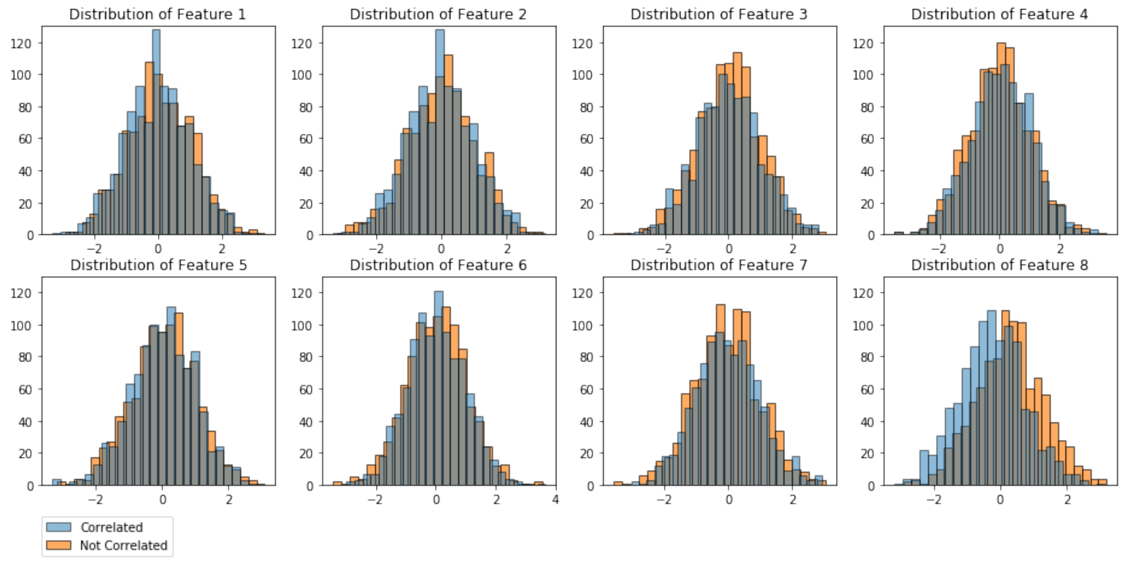


Figure 2-5. Histograms of the distribution of data in each feature, for $C_{F_1,F_2} = 1$ with F_8 shifted by 0.5.

F_1 and F_2 in the non correlated class), it could be argued that F_1, F_2, or any of the other six features should be dropped due to being irrelevant, **as there is no correlation between any single feature and it's class label**.

The only exception to this is in a secondary experiment where F_8 is sampled from a mean of 0.25 in the non-correlated class and -0.25 in the correlated class. This is done to analyze how a model might leverage correlation in the presence of a (slightly) separable feature.

2.1.1.1. Implementation

In terms of actual implementation, we utilize a multivariate normal with a specified covariance matrix (a matrix indicating which feature sets are correlated and how strong the correlation is). The covariance matrix allows us to generate data correlated amongst as many feature subsets as we like.

Listing 2.1 Code to generate a sample dataset for desired correlations

```
def synthetic_data(N,F,U,Shift_U ,Shift_I ,Pairs ,Corrs ):
"""
Generate synthetic data based on correlations pairs/coeffs
N : Number of data points to generate.
F : Number of features in these data points
U : Mean of distributions
Shift_U : Total shift of index Shift_I
Shift_I : Index to shift +- Shift_U/2, note Shift_I < F
Pairs : List of list of features to correlate (Feat 1 - F)
Corrs : Correlation coefficient for each set of pairs.
"""
# Not correlated
m_0 = np.ones(8)*U; m_0[Shift_I] += Shift_U/2
```

```
c_0 = np.eye(F)

# Correlated
m_1 = m_0.copy(); m_1[Shift_I] -= Shift_U
c_1 = np.eye(F)
for s,p in enumerate(Pairs):
for i in range(len(p)):
for j in range(i+1,len(p)):
c_1[p[i]-1][p[j]-1] = Corrs[s]
c_1[p[j]-1][p[i]-1] = Corrs[s]

# X0,X1 are N by F, X is 2N by F
X0 = np.random.multivariate_normal(m_0, c_0, N)
X1 = np.random.multivariate_normal(m_1, c_1, N)
X  = np.concatenate([X0,X1])
Y  = np.concatenate([np.zeros(N), np.ones(N)])
return X,Y
```

2.1.2. Training and Testing

The goal of the experiment is to determine if a model can leverage correlations, and if it can, how it does so. To test this a series of models were use. In depth explorations to the models' mathematical abilities or failures to utilize correlations are given in Appendix Machine Learning Models. We examine correlations with respect to feature selection (in which correlated features are commonly removed) and in classification algorithms. Six different models were examined:

- Random Forest
- Multilayer Perceptron
- Support Vector Machine
- K-Nearest Neighbors
- Logistic Regression
- Gaussian Naive Bayes

Both logistic regression and naive Bayes were considered as baseline algorithms, as they should be unaffected by correlations, due to assumptions of independence among features.

To test each algorithm's use of correlations, data was generated with F_1 and F_2 correlated with $C_{F_1,F_2} = 0, 0.1, 0.2, \ldots, 0.9, 1.0$ for a total of 11 values. At first no shift was applied to the means of any feature, making the data completely inseparable with respect to any *single* feature.

The following process was repeated for each C_{F_1,F_2}, Algorithm 1. Note that **MODELS** refers to a list model objects inheriting methods *fit* and *score* from a parent. The two parameters R and I control the outer loop (number of train/testing splits) and the inner loop (repeats of each algorithm). The inner loop parameter is specified due to some algorithms (notably the multilayer perceptron and random forest) having random aspects to their training process. This is discussed more in the Design of Experiments, Section 2.2.

Algorithm 1 Training/Testing Loop

```
1: for $r := 1$ to $R$ do
2:     $X_{\text{train}}, X_{\text{test}}, Y_{\text{train}}, Y_{\text{test}}$ =train_test_split($X, Y$)
3:     for $m$ in MODELS do
4:         for $i := 1$ to $I[m]$ do
5:             $m$.fit($X_{\text{train}}, Y_{\text{train}}$)
6:             accuracy[$m$][$r$][$i$] = $m$.score($X_{\text{test}}, Y_{\text{test}}$)
7:             f1[$m$][$r$][$i$] = f1_score($Y_{\text{test}}$, $m$.predict($X_{\text{test}}$))
```

It is quite apparent that this training/testing paradigm is *embarrassingly parallel*. Even creating workers to distribute the inner runs for the multilayer perceptron and random forest provided a significant speed up for moderate $I \sim 40$. Workers could also be created for the entire process, parallelizing over each C_{F_1,F_2} value. This is especially efficient if the number of processors available exceeds the number of correlation values to test.

2.1.2.1. Experiment Output

As seen in the psuedocode, the output from each training/test iteration is the classification accuracy on the test set and the F1 score. These values are recorded and grouped by the correlation value C_{F_1,F_2} for the dataset they came from. This allows a comparison of the two variables amongst all correlation values for each model.

2.1.2.2. Implementation

Using SKLearn's models the training/testing implementation is relatively simple. Two functions `print_header` and `pfi` are not defined here, but they print the header seen Listing 2.2 and display the permutation feature importance bar chart as seen in Figure 2-6 on page 32. The class `Results` is a data structure to store results by algorithm, with a few helper methods. The code is given in Listing 2.3.

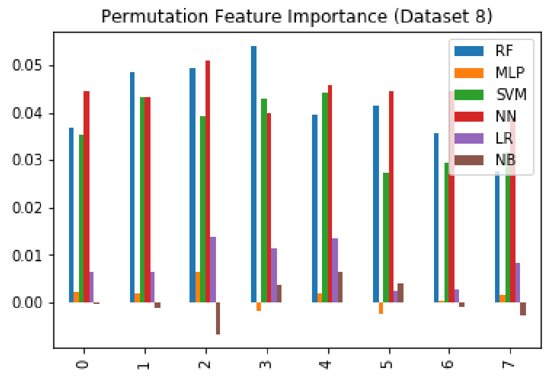
The output from Listing 2.3 is given in Listing 2.2. The inner and outer loop parameters are $R = 1$ for a single training/test split of the data, and $I = 10$ for the random forest and multilayer perceptron and $I = 1$ for the remaining (non-random) models. Examples of the permutation feature importance plots are given in Figure 2-6 on page 32, for $C_{F_1,F_2} = 0.2, 0.5, 0.8, 1.0$.

Listing 2.2 Output from Listing 2.3 for a single correlation value.

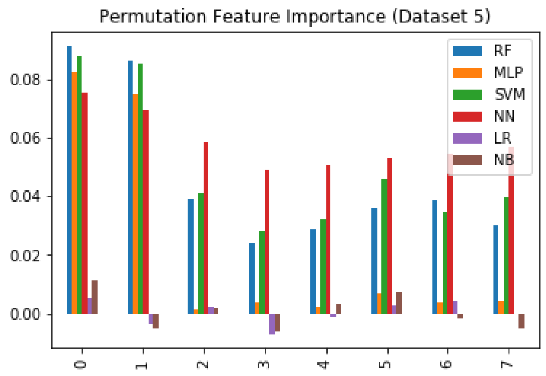
```
Data Set 0
Correlations:
[1, 2]      :     1.000000
Shifted U: 0.000 Standard Deviations
Shifted Feature(1-8): 8
========================
Outer: 1
=========
Model: RF       (10)    Train: 0.891  Test: 0.696  F1: 0.702  Time: 1.790
Model: MLP      (10)    Train: 0.972  Test: 0.958  F1: 0.961  Time: 21.889
Model: SVM      (1)     Train: 0.823  Test: 0.715  F1: 0.748  Time: 0.254
Model: K-NN     (1)     Train: 0.849  Test: 0.655  F1: 0.683  Time: 0.086
Model: LR       (1)     Train: 0.549  Test: 0.502  F1: 0.492  Time: 0.003
Model: NB       (1)     Train: 0.545  Test: 0.468  F1: 0.471  Time: 0.002
============================================================================
Permutation Feature Importance:
0         1         2         3         4         5         6         7
RF    0.26510   0.26635   0.03690   0.02835   0.03640   0.02740   0.03160   0.03035
MLP   0.46935   0.46685   0.00080   0.00010  -0.00060  -0.00030   0.00055  -0.00095
SVM   0.24495   0.24535   0.03325   0.02215   0.02420   0.02905   0.02980   0.02730
NN    0.18600   0.17440   0.04585   0.04490   0.04405   0.04660   0.04820   0.03615
LR    0.00765   0.00495   0.02945  -0.00065   0.00700   0.00700   0.00830   0.00110
NB   -0.00845  -0.00515   0.00715   0.00070  -0.00170  -0.00250   0.00165   0.00665
```

Listing 2.3 Code to run the training/test loop as outlined in Algorithm 1

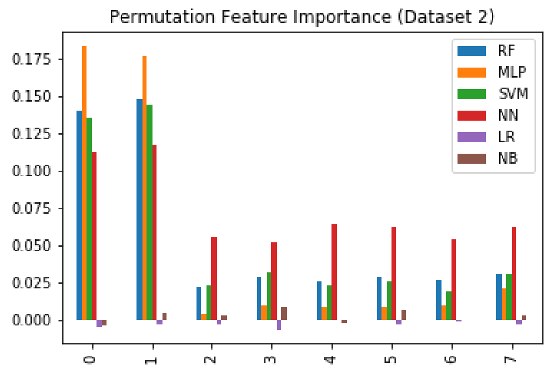
```
ALGS = ['Random␣Forest', 'Multilayer␣Perceptron', 'Support␣Vector␣Machine',
        'K-Nearest␣Neighbor', 'Logistic␣Regression', 'Naive␣Bayes']
REPS = [10, 10, 1, 1, 1, 1]

rf = RandomForestClassifier(max_depth=5)
mlp = MLPClassifier(hidden_layer_sizes=(5), max_iter=2000)
svm = SVC(gamma='auto', probability=True)
nn = KNeighborsClassifier(n_neighbors=3)
lr = LogisticRegression()
nb = GaussianNB()

models = [rf,mlp,svm,nn,lr,nb]
RES = []
for c in range(C):
    output = Results(C_Pairs,C_Corrs[:,c],ALGS)
    print_header(c,C_Pairs,C_Corrs,Shift_U,Shift_I)
    for r in range(R):
        print("Outer:␣{:d}\n{:}".format(r+1,"="*(9+int(np.log10(r+1)))))
        # Generate random 70/30 split
        X_train, X_test, y_train, y_test =
            train_test_split(XList[c],Y, test_size=0.3)
        # Test split on each model
        tot_t = []
        for i,model in enumerate(models):
            # Repeat testing for random algs
            sub_res = []
            tot_t.append(time.time())
```



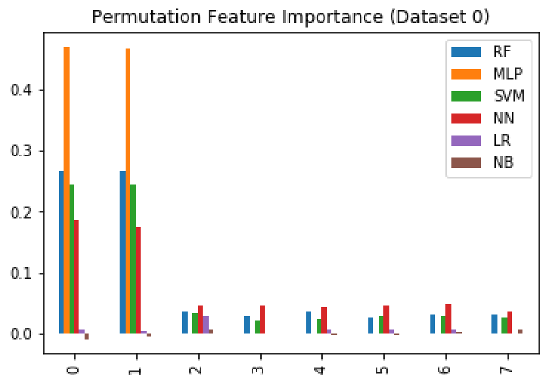
(a) Permutation feature importance for $C_{F_1,F_2} = 0.2$



(b) Permutation feature importance for $C_{F_1,F_2} = 0.5$



(c) Permutation feature importance for $C_{F_1,F_2} = 0.8$



(d) Permutation feature importance for $C_{F_1,F_2} = 1.0$

Figure 2-6. Permutation feature importance values for a variety of C_{F_1,F_2}. Note this is from a single experiment, results are subject to variation.

```
    for in_r in range(REPS[i]):
        model.fit(X_train, y_train)
        train_score = model.score(X_train,y_train)
        test_score = model.score(X_test,y_test)
        f1 = sklearn.metrics.
            f1_score(y_test,model.predict(X_test))
        sub_res.append((train_score,test_score,f1))
        tot_t[i] = time.time()-tot_t[i]
        output.record_alg(ALGS[i],sub_res)
        stats = reduce((lambda x,y: np.array(x)+y),
                sub_res)
        print("Model:␣{:<24}({:d})\tTrain:␣{:.3f}␣␣Test:␣{:.3f}␣␣F1:␣{:.3f
            }␣␣Time:␣{:.3f}"
        .format(ALGS[i],REPS[i],stats[0]/REPS[i],stats[1]/REPS[i],
        stats[2]/REPS[i],tot_t[i]))
        RES.append(output)
        print("="*81)
        pfi(models,XList[c],Y,c)
```

2.2. Design of Experiments

In order to determine whether or not a machine learning model has the ability to leverage the information found in higher-order interactions among features (such as correlations), several sources of variance need to be accounted for. This minimizes the likelihood that evidence of the use of correlations comes from random variance instead of actual correlations in the data.

For the first experiment our variable of interest is the strength of the correlation between F_1 and F_2. This has been systematically varied from 0 to 1 by 0.1, inclusive. This gives eleven sets of results that can be compared, to show how the accuracy and F1 score of a machine learning model vary under different strengths of correlation. Note that in this experiment correlation is the only metric of separation between the classes.

The second experiment adds a second variable of interest, whether a slight degree of separation between the classes in a single feature changes the ability of a model to leverage correlation. Following an identical procedure as the first experiment, a single feature (F_8) is given a mean of -0.25 for the correlated class and 0.25 for the non-correlated class. The correlation between F_1 and F_2 is still varied from 0 to 1 by the same increments. Thus we can compare this second set of results with the first to draw conclusions about the effects of slight separability on the usage of correlations as a method of separation.

The variables in the model that are controlled include:

- Feature distributions: All are drawn from a normal distribution with $\mu_i = 0$ and $\sigma_i = 1$ for $i = 1, 2, \ldots, 8$. In the second experiment μ_8 is deliberately modified.
- Training/Test proportion: A 70/30 split is always used.

The variables in the model that cannot be controlled:

- Training/Test split: data is randomly assigned to each category.
- Random algorithms: Notably random forests and multilayer perceptrons have random aspects to their training.

To control for the variance that may arise from the training/test split and the randomness of some of the algorithms we introduce a number of outer repeats (R) and inner repeats ($I[m]$ for each model m).

- R: The number of times to generate a training/test split.
- $I[m]$: The number of times to train/test each model m for each training test split.

Thus for each correlation value C_{F_1,F_2}, each model m is run $R \times I[m]$ times. We fix $I[m] = 1$ for the support vector machine, k-nearest neighbor, logistic regression, and naive Bayes models, as they have no random nature to their training (i.e. with the same training/test set the model will get the same score every time). This is slightly unfortunate, as the random forest and multilayer perceptron are two of the slower models to train/test.

We increase R and $I[m]$ to decrease the variance in the mean accuracy and F1 score of each model/correlation value. As well, this provides smoother distributions of results, painting a better picture of what the actual realizations of these experiments are.

2.3. Results

Two experiments were run. The first utilized correlations for F_1 and F_2 such that $C_{F_1,F_2} = 1.0, 0.9, \ldots, 0.1, 0.0$ with $\mu_i = 0$ for $i = 1, 2, \ldots, 8$ for both classes. This compares the accuracy and F1 score of a model to various levels of correlation.

A second experiment was run, identical to the first in every way, except for $\mu_8 = \pm 0.250$, with a negative mean for the correlated class and a positive mean for the non-correlated class. This allowed for comparing the accuracy and F1 score of a model as the correlation decreased, but a single-feature degree of separability remained the same.

Now, since these experiments control for almost all of the same variables it is apparent that each is interesting in the context of the other. Thus we present their results together as a means of comparing the trends over various C_{F_1,F_2} with and without a shifted feature.

The design of experiments for this problem consisted of $R = 1000$ for twenty training/test splits. The inner repeats for the random forest and multilayer perceptron were $I[m] = 10$. This yielded 10,000 runs for each of those algorithms with 1000 runs for the deterministic ones. Note that these runs are for each C_{F_1,F_2} value and for each shifted/non-shifted data set.

With respect to the feature shift, the mean of one class is 0.5 standard deviations from the other (both have identical standard deviations). Since a normal distribution was used to generate these samples, approximately 20% of the data for each distribution should be drawn from this area. Using properties of this distribution we can get an idea of how separable this data is, with respect to just F_8.

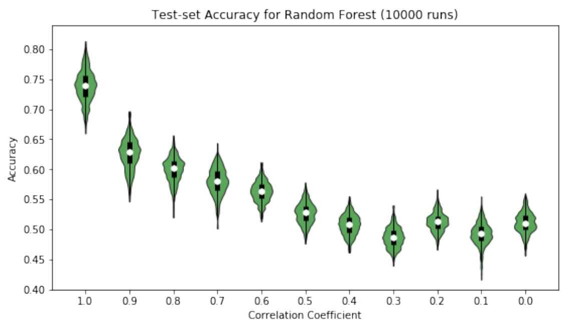
From the perspective of the data drawn from $\mu = -0.25$ we summarize how much data should lie in various ranges in Table 2-1. If we draw a line at 0 and attribute anything less to the class with $\mu = -0.25$ and anything more to the class with $\mu = 0.25$, we should end up correctly identifying the class of approximately 60% of the data.

Table 2-1. Distribution of F_8 for $\mu = -0.25$.

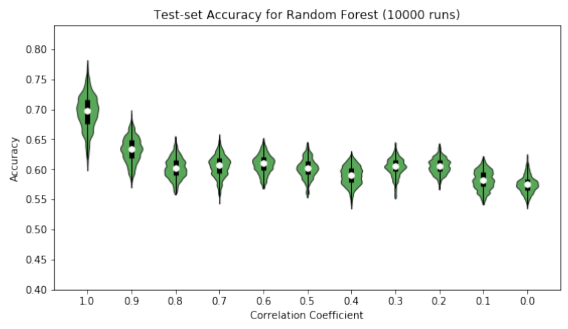
Range	Proportion of Data	Class attributed to
$[-\infty, -0.25]$	0.50	-
$[-0.25, 0]$	0.099	-
$[0, 0.25]$	0.093	+
$[0.25, \infty]$	0.308	+

2.3.1. Random Forest

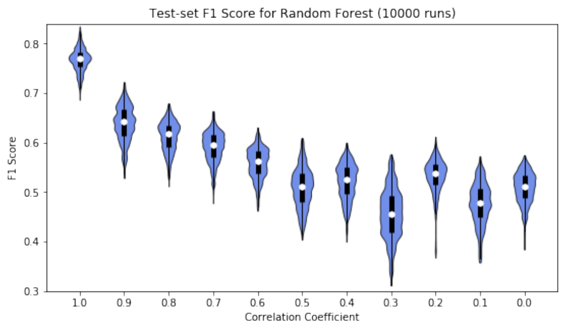
In the scope of the first experiment we see a well defined curve of decreasing accuracy and F1 score as the correlation between F_1 and F_2 drops. This supports our hypothesis, as we can plainly see there is some information in the correlation that is being leveraged to separate the classes. This degree of separation is connected to the strength of the correlation.



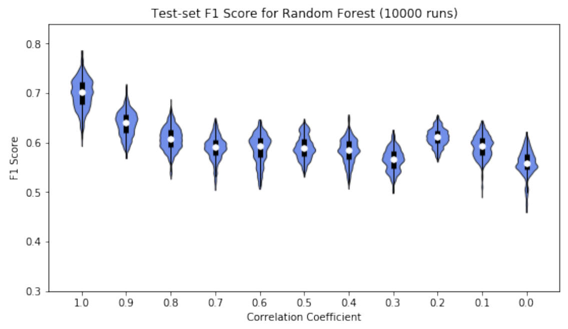
(a) Accuracy for random forest, no feature shift.



(b) Accuracy for random forest, with a feature shift.



(c) F1 score for random forest, no feature shift.



(d) F1 score for random forest, with a feature shift.

Figure 2-7. Violin plots of accuracy and F1 score for random forests over each C_{F_1,F_2} value (10,000 runs per C value).

In the scope of the second experiment we note a few interesting points. Most notably, the accuracy for $C_{F_1,F_2} = 1.0$ was actually lower than in the non-shifted data. In Appendix Random Forest we discuss the properties of a random forest that allow it to recognize correlations at length, but put succintly, this is likely due to node splits on F_8 taking priority over shifts on both correlated features. As with the first experiment's results, we see a decrease in accuracy/F1 score as the strength of the correlation decreases. However, this seems to be baselined by an approximate accuracy of 0.6 provided by the separation in F_8.

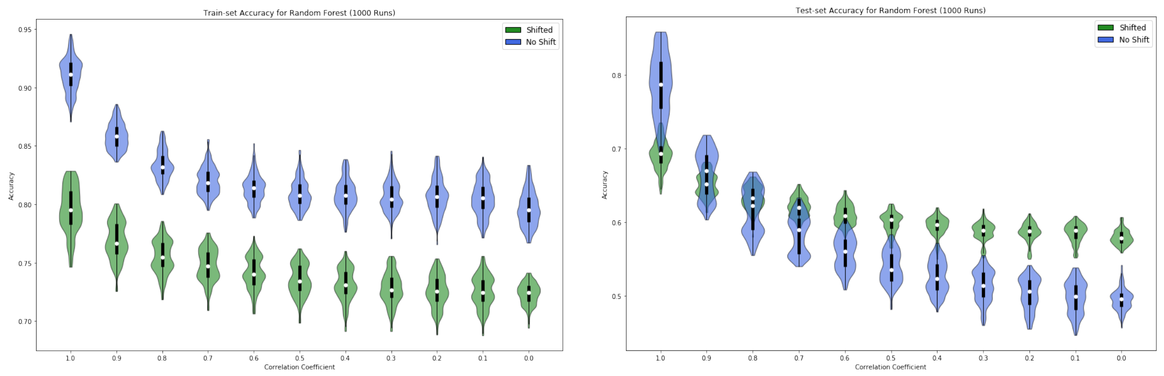
This baseline of 0.6 lines up with the results from Table 2-1 on the preceding page, where a simple plus/minus classifier on F_8 would correctly attribute approximately 60% of the data. This type of simple classifier makes sense in the context of how a random forest works, as F_8 would be best split near 0 and the remaining features are just noise.

2.3.1.1. Training/Test Analysis

In a secondary experiment, we analyze an interesting pattern found in the training and testing accuracies of the random forest. This experiment follows all the same parameters as the original experiments, except for only random forests were studied and a different number of runs were used. Data at each respective correlation was generated 10 times, each of those data sets having

10 training/test splits, each with 10 random forests generated for a total of 1000 runs per correlation value.

With the results from this experiment in Figure 2-8 we see that although the non-shifted data has a higher training set accuracy than the shifted, at $C_{F_1,F_2} \leq 0.8$ we see that the relative position of the two datasets change under the test set accuracy. In the test set, the accuracy score of the non-shifted data drops much faster than that of the shifted data.



(a) Train-set accuracy for random forest. **(b)** Test-set accuracy for random forest.

Figure 2-8. Violin plots of training/test accuracy for random forests over each C_{F_1,F_2} value (1000 runs per C value).

This rapid drop in test-set accuracy for the non-shifted data comes from how the random forest "sees" correlations. Rather than understanding that $F_1 \approx F_2$ for high correlation, it understands that the areas away from $F_1 = F_2$ are likely to belong to the non-correlated class. The areas that a single decision tree assigns are hyper-rectangles parallel to the axis. This means that in training they establish vertices near $F_1 \approx F_2$ to partition off non-correlated areas. As the correlation decreases, and F_1 and F_2 vary more and more from each other, it is likely that over-fitting occurs on the correlated points furthest from $F_1 = F_2$, leading to a decrease in accuracy. We can visualize this, using a rectangle to indicate an area in the training data consisting of entirely non-correlated points. In Figure 2-9 on the next page we see that in the case of $C_{F_1,F_2} = 1$ (left) that it would be impossible for a test point to exist in the shaded area. However, on the right, with $C_{F_1,F_2} = 0.8$, it is entirely possible, that test point(s) exist in the shaded area, leading to a lower test set accuracy. Since random forests use decision trees that have boundaries of this form, it is likely that this explains the over-fitting seen in the results.

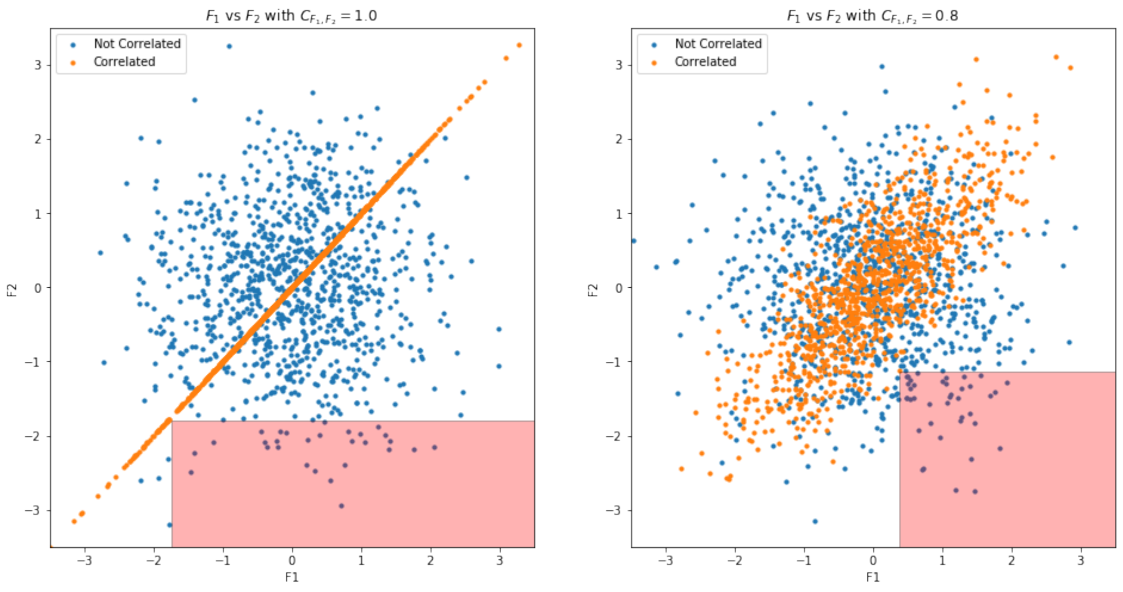


Figure 2-9. Comparison of data distribution for $C_{F_1,F_2} = 1.0$ and $C_{F_1,F_2} = 0.8$ with hypothetical decision tree shading on the training set.

2.3.1.2. Random Forest Feature Importance

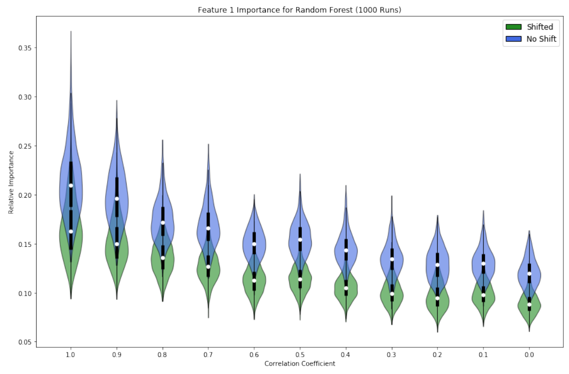
Random forests have a built-in measure of feature importance. In `sklearn` these are implemented as impurity based, as the normalized total reduction of the GINI Impurity by a given feature. There are many caveats to using these types of importance measures, especially when it comes to correlations, but if they are analyzed under the context in which they are designed they can provide faithful insight.

In the same experiment as the previous section, the feature importance was tracked over the 1,000 runs per correlation value. The results are given in Figure 2-10 on the following page, for features F_1, F_2, F_3, and F_8, which are the two correlated features, a noise feature, and the shifted feature, respectively.

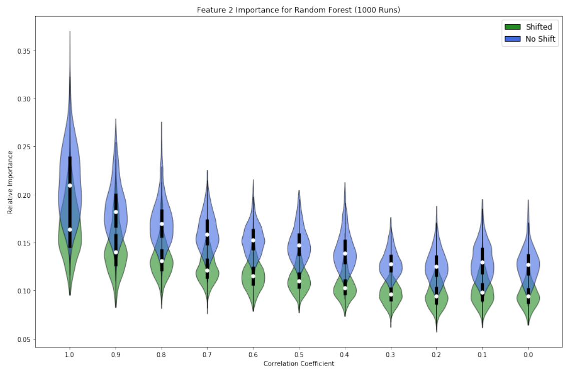
From this we see that F_1 and F_2 are of relatively equal importance. This makes sense, as these features need to be used in tandom to provide separability.

For F_3 we see a growing importance as correlation decreases, though noting the scale it is not a large increase. This is due to the distribution of the decrease in importance of F_1 and F_2 (as correlation decreases) amongst the remaining features (this is a normalized value). Note that even though only F_3 is displayed, F_4 to F_7 will have similar results (they are all noise features).

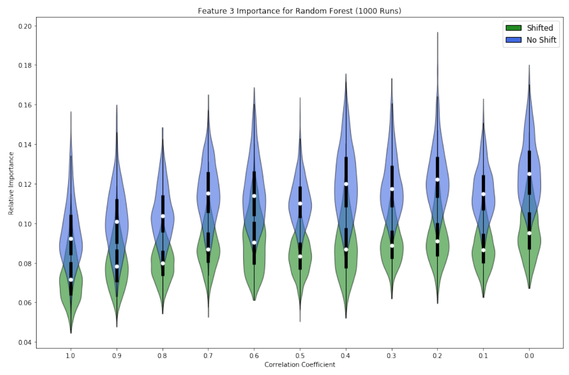
Finally in F_8 we see similar results for the non-shifted data, and a very high importance for the shifted data. The increase in importance as the correlation decreases has the same explanation as that for F_3.



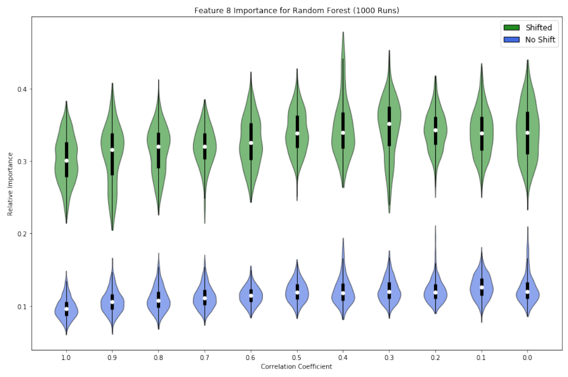
(a) Random forest feature importance, F_1



(b) Random forest feature importance, F_2



(c) Random forest feature importance, F_3



(d) Random forest feature importance, F_8

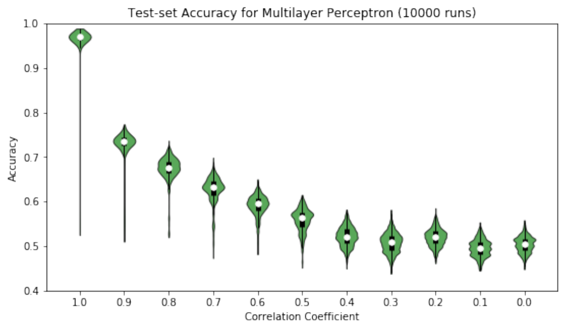
Figure 2-10. Violin plots of feature importance for F_1, F_2, F_3, F_8 (left to right, top to bottom) for all C_{F_1,F_2} values (1,000 runs per C value).

2.3.2. Multilayer Perceptron

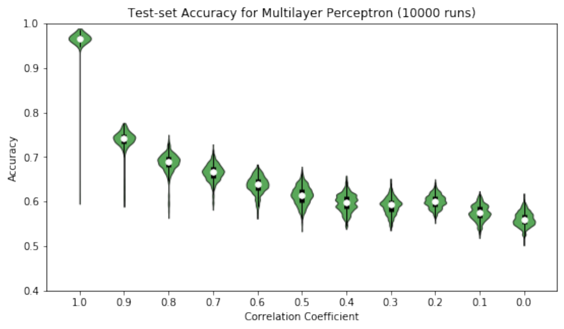
It should be noted that the long tails seen on these plots indicate an instance in which the multilayer perceptron completely diverged during training. What is interesting about this breakdown of the network is that it seems to be in the neighborhood of 0.5 for the first experiment (data without a shifted feature). This makes sense, as if the model is "randomly" assigning output without respect to the correlation, or always assigning one class, it should have an accuracy of near 0.5. In the second experiment, with the shifted feature, the lower tail seems to stop in the neighborhood of 0.55-0.60. This higher-accuracy lower tail indicates that the network has failed to read a correlation in the data, and is simply separating the data on F_8, which has a basic separability accuracy of 0.6, as seen in our results in Table 2-1 on page 34.

With regards to the first experiment, we see very clear exponential growth of the accuracy of the model as the strength of the correlation between F_1 and F_2 increases. This is repeated in the F1 score for the first experiment, where we also see a growing variance in the F1 score as the correlation decreases. This is extremely strong evidence for the ability of a neural network to utilize the presence of a correlation in the lack of any individual feature separability.

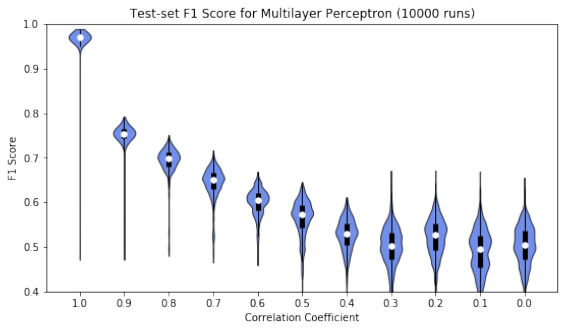
With the second experiment we see a result similar to the random forest. There exists a baseline,



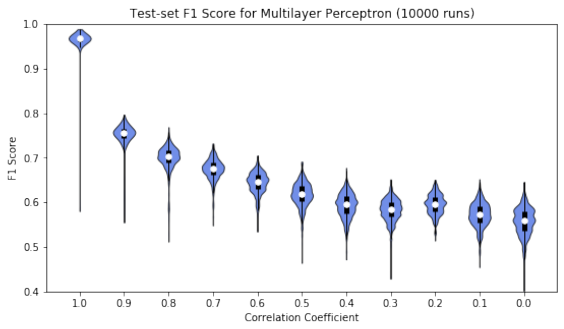
(a) Accuracy for multilayer perceptron, no feature shift.



(b) Accuracy for multilayer perceptron, with a feature shift.



(c) F1 score for multilayer perceptron, no feature shift.



(d) F1 score for multilayer perceptron, with a feature shift.

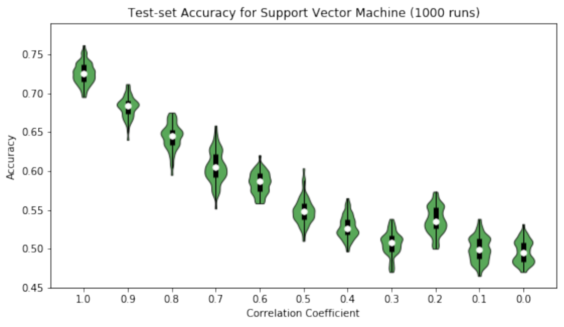
Figure 2-11. Violin plots of accuracy and F1 score for multilayer perceptrons over each C_{F_1,F_2} value (10,000 runs per C value).

similar to the one previously mentioned, which stops the network from decreasing to "random predictions" (accuracy of 0.5). This comes from the slight separability of F_8 and falls in line with the discussion of a basic classifier scoring $\sim 60\%$ on that data.

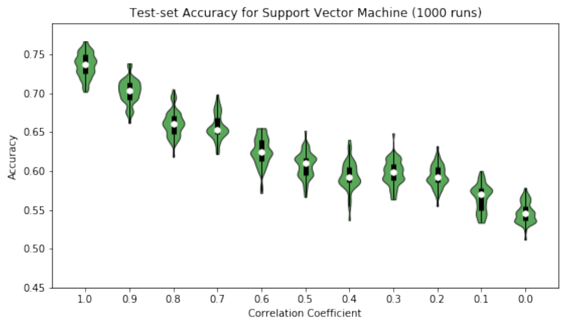
2.3.3. Support Vector Machine

With these results we see a relatively smooth decrease as the correlation drops. Similarly to the previous models, we notice that the non-shifted data decreases down to an accuracy to 0.5 as expected, while the shifted decreases to somewhere between 0.55 and 0.60. Again, this is due to the natural separability of around 0.6 in the shifted data (Table 2-1 on page 34).

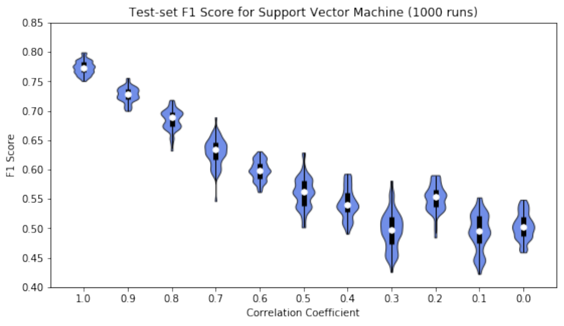
It is interesting to note that results for the shifted and non-shifted data have a similar accuracy for $C_{F_1,F_2} = 1.0$. This indicates that the algorithm might only be able to rely on one of the two means of separability at a time. Alternatively, it may be that in either case of shifted or non-shifted that the data is only separable to 0.75 accuracy with a support vector machine. In Appendix Support Vector Machine this is explored and it is found that the shift does increase the accuracy for $C_{F_1,F_2} = 1$, it is just not very noticeable in these results.



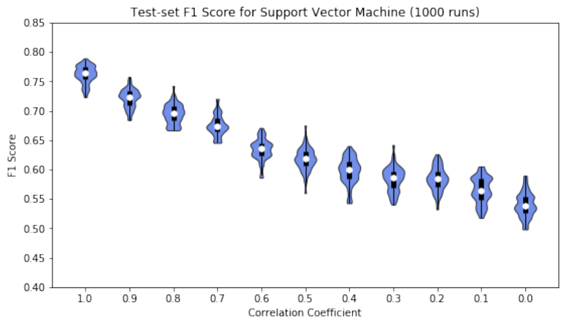
(a) Accuracy for support vector machine, no feature shift.



(b) Accuracy for support vector machine, with a feature shift.



(c) F1 score for support vector machine, no feature shift.



(d) F1 score for support vector machine, with a feature shift.

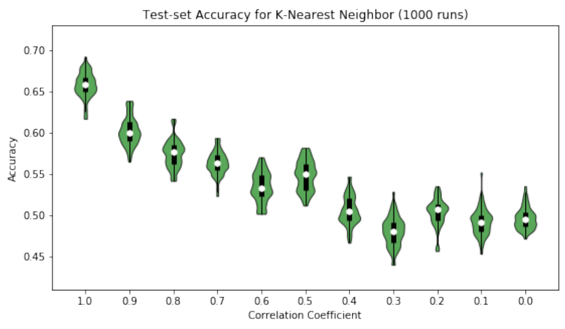
Figure 2-12. Violin plots of accuracy and F1 score for support vector machines over each C_{F_1,F_2} value (1,000 runs per C value).

2.3.4. K-Nearest Neighbor

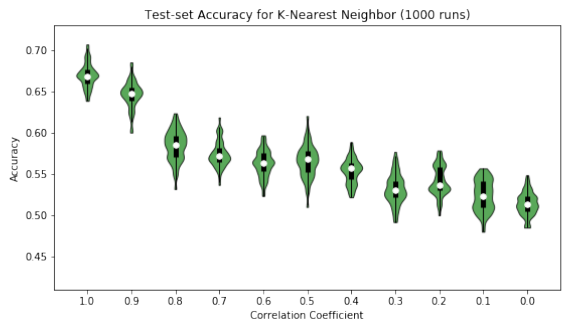
With these results we see a significant drop off in accuracy and F1 score as the correlation decreases. For the non-shifted data, this comes between $C_{F_1,F_2} = 0.9$ and $C_{F_1,F_2} = 0.8$, where the decrease in accuracy holds relatively constant until $C_{F_1,F_2} = 0.5$. We see the accuracy bottom out near 50%, indicating that there is no information that the model can leverage to separate the classes.

For the shifted data, we see the results for $C_{F_1,F_2} = 1.0$ and $C_{F_1,F_2} = 0.9$ are nearly identical to that of the non-shifted. The major difference comes at the lower limit, where for little or no correlation we see accuracy near 55%. This can most likely be attributed to the factors given previously in the discussion about the purely statistical separability of the slight feature shift.

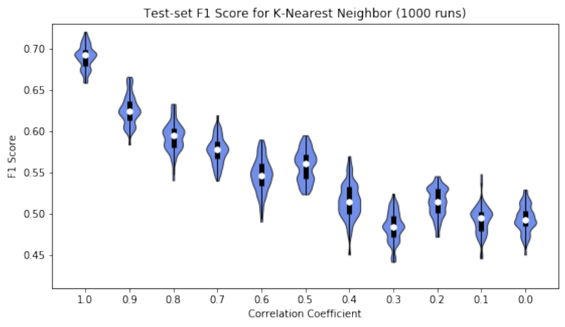
From these results it is apparent that the K-Nearest Neighbor algorithm has the ability to leverage correlations in the data. However, it seems that this ability stems from the higher-order distribution of the data rather than the workings of the algorithm itself. This is discussed at length in Appendix K-Nearest Neighbor.



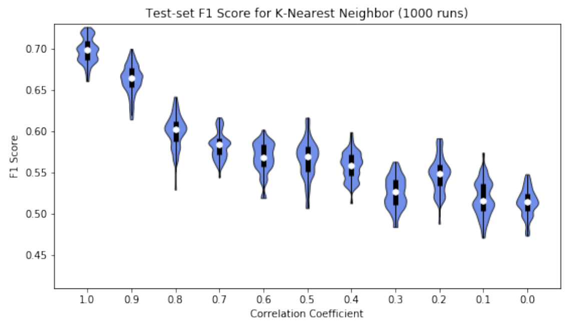
(a) Accuracy for k-nearest neighbor, no feature shift.



(b) Accuracy for k-nearest neighbor, with a feature shift.



(c) F1 score for k-nearest neighbor, no feature shift.



(d) F1 score for k-nearest neighbor, with a feature shift.

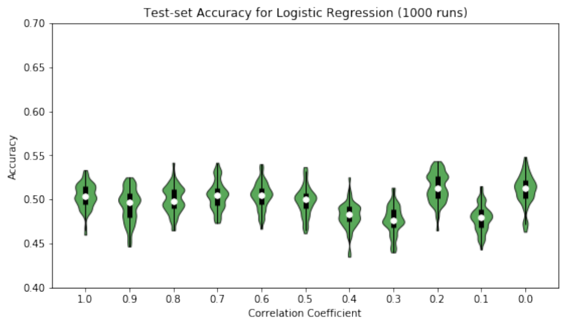
Figure 2-13. Violin plots of accuracy and F1 score for k-nearest neighbors over each C_{F_1,F_2} value (1,000 runs per C value).

2.3.5. Logistic Regression

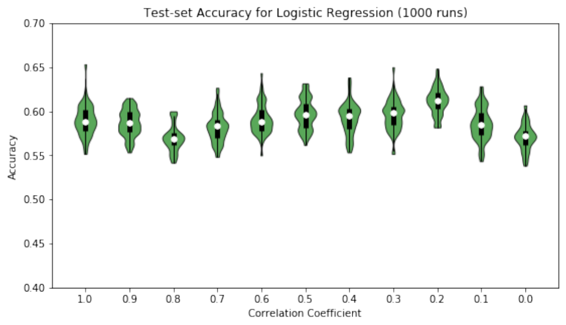
With logistic regression we see results that line up with the discussion of separability on F_8. The non-shifted data has average accuracies that fluctuate around 0.5, regardless of the correlation strength, indicating that the algorithm is assigning classes randomly. In the case of the shifted data, this fluctuation is approximately 0.6, which goes back to the theoretical separability of shifted F_8 as discussed in Table 2-1 on page 34.

As for how the correlation strength affects the results, both the shifted and non-shifted results are independent of the correlation. This comes from the assumption that the features are independent that is a part of logistic regression.

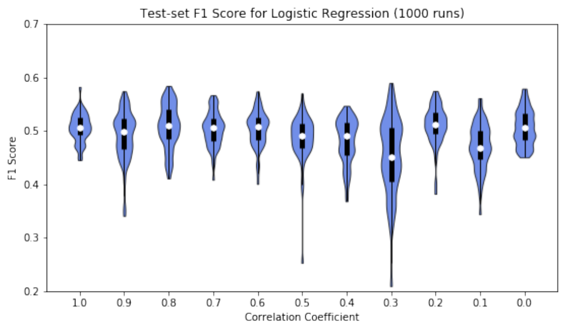
In terms of the actual mathematically machinery that might see correlation, logistic regression essentially uses a linear separator with a "wave" of probability that increases/decreases away from $P(x) = 0.5$ as you move away from the separator. The rate of this change is determined by the magnitude of the linear separator. Since the non-shifted data has no separation with respect to any single feature, there is not anywhere that one can "draw" a linear separator and achieve any meaningful accuracy. In the case of the shifted data, the separator lies near $F_8 = 0$, to separate on F_8, while the separation with respect to the rest of the features is meaningless. This is further explored in Appendix Logistic Regression.



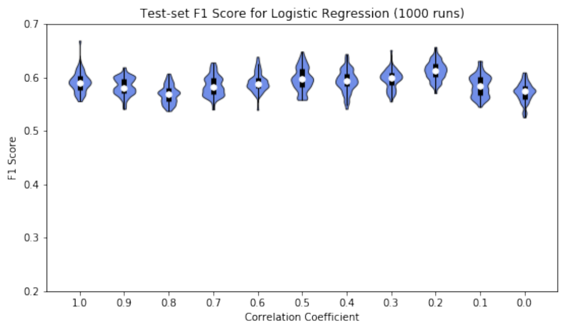
(a) Accuracy for logistic regression, no feature shift.



(b) Accuracy for logistic regression, with a feature shift.



(c) F1 score for logistic regression, no feature shift.



(d) F1 score for logistic regression, with a feature shift.

Figure 2-14. Violin plots of accuracy and F1 score for logistic regression over each C_{F_1,F_2} value (1,000 runs per C value).

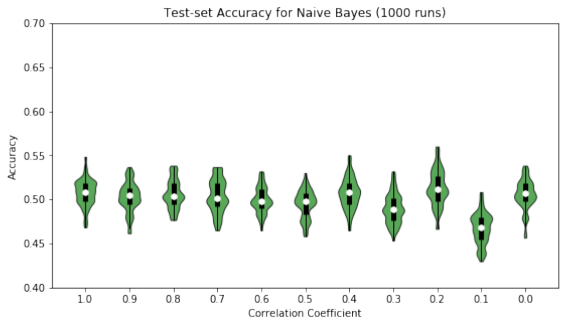
2.3.6. Gaussian Naive Bayes

With Gaussian Naive Bayes we see results similar to that of logistic regression. In the case of the non-shifted data the classifications are essentially random. In the shifted data we see the results hovering near 0.6, which is the theoretical separability of F_8, from the discussion around Table 2-1 on page 34.

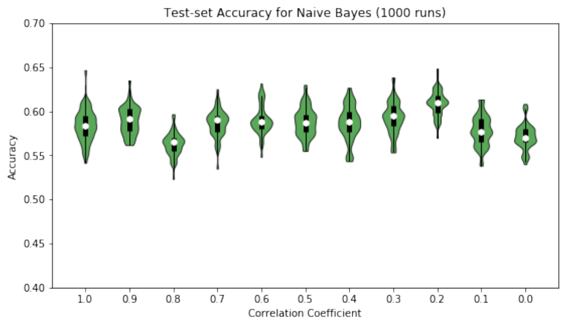
In terms of change with respect to correlation value, there is none. This is due to the glaring assumption of Naive Bayes that all features are independent (this is given in the derivation of the method using Bayes' theorem). Thus there is no mechanics for the algorithm to understand separability based on a joint distribution of F_1 and F_2.

2.4. Feature Selection Algorithms

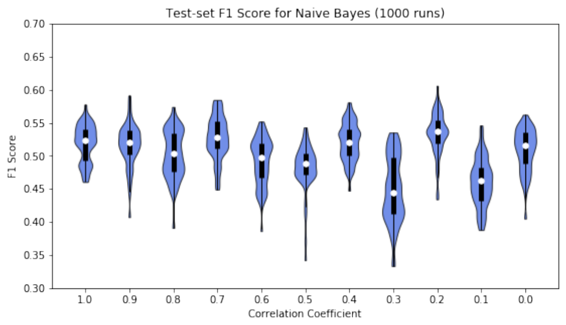
Dimension reduction proposes an increase in learning performance, computational efficiency, and model generalization while offering a decrease in memory storage [65]. This can be achieved through feature extraction (combining features) or feature selection (choosing a subset). We focus our discussion around feature selection as most MLE methods choose a subset of the most influential features on a prediction.



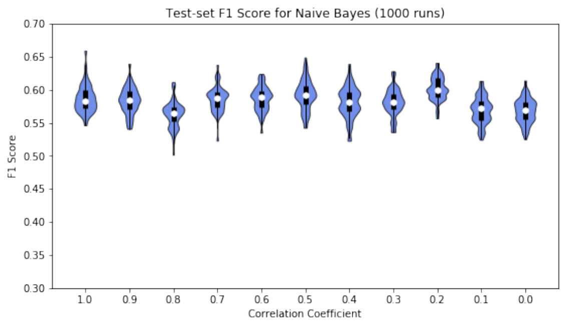
(a) Accuracy for naive Bayes, no feature shift.



(b) Accuracy for naive Bayes, with a feature shift.



(c) F1 score for naive Bayes, no feature shift.



(d) F1 score for naive Bayes, with a feature shift.

Figure 2-15. Violin plots of accuracy and F1 score for naive Bayes over each C_{F_1,F_2} value (1,000 runs per C value).

There is a wealth of information surrounding feature selection. A review by the authors in [65] divides feature selection algorithms into four categories:

1. **Similarity-Based Methods**: These methods assess a feature's importance as its ability to preserve data similarity, either by label information or distance measures in the data.

2. **Information-Theoretical-Based Methods**: These methods attempt to maximize feature relevance (feature-class correlation) and minimize redundancy (feature-feature correlation).

3. **Sparse-Learning-Based Methods**: These methods attempt to minimize the fitting errors while regularizing towards smaller or zero values in features which can then be eliminated (i.e. Lasso regression).

4. **Statistical-Based Methods**: These methods rely on statistical measures to to determine feature relevance (typically filter based).

We implement a few feature selection algorithms and analyze their performance on our overlapping, correlated data set 2.1.1.

2.4.1. Relief

Relief selects only statistically relevant features, with respect to a binary classification problem [55]. It employs a "filter-method approach to feature selection that is notably sensitive to feature interactions". In general, it accomplishes this by the idea of a near-miss and near-hit in the neighborhood of a given instance, and then updating relative feature weights based on a distance measure to these respective points.

More specifically, the algorithm selects a random instance x_i in the data and then randomly picks a nearby positive (same class) instance x_{pos} and negative (different class) instance x_{neg} from k of the nearest neighbors. If x_i is positive, then x_{pos} is a near-hit, and x_{neg} is a near-miss. A weight vector W over the features is then updated so that:

$$w = w - (x_i - x_{pos})^2 + (x_i - x_{neg})^2 \tag{2.1}$$

This process repeats m times and then the weight vector w is averaged. Features that are above some threshold τ are considered important.

Essentially, feature values that are close between x_i and a near-hit x_{pos} and distant between x_i and a near miss x_{neg} have strong weights. Features that have a large distance between x_i and a near-hit or a small gap between x_i and a near-miss are weighted less. A practical implementation is given in Listing 2.4.

Running the algorithm with $m = 500$ and $k = 40$ on a dataset consisting of eight features, all drawn from a normal distribution with $\mu = 0$ and $\sigma = 1$, where a binary classification is given by a correlation of $C_{F_1,F_2} = 1$ for features one and two yields a weight vector

$$W = \begin{bmatrix} -0.075 & -0.068 & 0.010 & -0.005 & 0.053 & -0.100 & -0.212 & 0.021 \end{bmatrix} \tag{2.2}$$

No single feature stands out as important, noting that the sign of the feature weight is important. However, this changes under a modified version of the algorithm, ReliefF, discussed in the next section.

Listing 2.4 Relief algorithm implementation.

```
def Relief(X,Y,m, tau ,nn):
"""
X:      np.array: instances by features
Y:      np.array: instances labels
m:      int     : number of repeats
tau:    float   : threshold for "important" features
k:    int     : number of nearest neighbors to consider for near selection
"""
X_pos = X[Y == 1,:]
X_neg = X[Y == 0,:]
n,p = X.shape
w = np.zeros(p)
for j in range(m):
        inst = np.random.randint(0,n)
        # Find a nearby positive instance Z+
        pos_list = np.argsort(np.sum((X_pos - X[inst])**2, axis=1))
        pos_inst = np.random.randint(0,k)
        if pos_inst == 0 and (X[inst,:] == X_pos[pos_list[0],:]).all():
                pos_inst = np.random.randint(1,k)

        # Find a nearby negative instance Z-
        neg_list = np.argsort(np.sum((X_neg - X[inst])**2, axis=1))
        neg_inst = np.random.randint(0,k)
        if neg_inst == 0 and (X[inst,:] == X_neg[neg_list[0],:]).all():
                neg_inst = np.random.randint(1,k)

        # Update W
        if Y[inst] == 1:
                w = w - np.square(X[inst,:]-X_pos[pos_inst,:]) +
                    np.square(X[inst,:]-X_neg[neg_inst,:])
        else:
                w = w - np.square(X[inst,:]-X_neg[neg_inst,:]) +
                    np.square(X[inst,:]-X_pos[pos_inst,:])

return (1/m)*w >= tau
```

2.4.2. ReliefF

A modified version of Relief, ReliefF, is proposed in [58], however it is better put in [96]. ReliefF operates almost identically to Relief; rather than randomly selecting a single near-miss and near-hit from k nearest neighbors, all of the k nearest neighbors are used. The update step for the weight matrix W is also different. The following is looped over each feature f:

$$w_f = w_f - \sum_{j=1}^{k} \text{diff}(x_{i,f}, h_{j,f})/(m \cdot k) + \sum_{C \neq \text{class}(x_i)} \left[\frac{P(C)}{1 - P(\text{class}(x_i))} \sum_{j=1}^{k} \text{diff}(x_{i,f}, q_{j,f}(C)) \right] /(m \cdot k) \tag{2.3}$$

where $x_{i,f}$ is the value of feature f for the x_i, $h_{j,f}$ is the value of f for the j^{th} instance from the set of near-hits and $q_{j,f}$ is value of f for the j^{th} instance from the set of near-misses. The probabilities $P(C)$ are determined by class frequency for class C, however for a binary class case with equal distribution we have that $P(C)/(1 - P(\text{class}(x_i))) = 1$. As before, m is the number of repeats. The diff function:

$$\text{diff}(x_{i,f}, I_{j,f}) = \frac{|x_{i,f} - x_{j,f}|}{\max(f) - \min(f)} \tag{2.4}$$

Where the max and min of f are the maximum and minimum values that feature f takes over the dataset.

In terms of vector math, Equation 2.3 can be rewritten as:

$$w = w - \frac{1}{mk} \frac{\sum_{j=1}^{k} |h_j - x_i|}{\vec{d}} + \frac{1}{mk} \frac{\sum_{j=1}^{k} |q_j - x_i|}{\vec{d}} \tag{2.5}$$

where $\vec{d} = \max(x_f \in X) - \min(x_f \in X)$ for each f in the set of features. Note the division by vector $\vec{d}$ is point-wise.

The ideas of penalizing far distances in features among near-hits and short distances in features among near-misses while rewarding short distances in features among near-hits and far distances among features in near-misses remains. A practical implementation is given in Listing 2.5.

Running the algorithm with $m = 500$ and $k = 20$ on the same data as in Relief (eight features all drawn from a normal distribution with $\mu = 0$ and $\sigma = 1$ where binary class labels are assigned by a correlation of $C_{F_1,F_2} = 1$ for features one and two) yields a weight vector

$$W = \begin{bmatrix} 0.020 & 0.020 & -0.0004 & -0.002 & -0.002 & -0.001 & -0.002 & -0.0001 \end{bmatrix} \tag{2.6}$$

Although the scores for F_1 and F_2 are not large, they are enough to stand out from the remaining scores (all less than zero and extremely small in magnitude). It is likely this "importance" of these two features comes from the increased density among F_1 and F_2 for the correlated class. This is discussed at length in Appendix A.1.3.

As a point of comparison, when the same parameters are used in ReliefF and the correlation for features one and two is set to zero, the weights for F_1 and F_2 drop to magnitudes near 0.001, while the magnitudes of the other features do not change.

Listing 2.5 ReliefF algorithm implementation.

```
def ReliefF(X,Y,m,nn):
"""
X:      np.array: instances by features
Y:      np.array: instances labels
m:      int     : number of repeats
nn:     int     : number of nearest neighbors to consider
"""
X_pos = X[Y == 1,:]
X_neg = X[Y == 0,:]
n,p = X.shape
w = np.zeros(p)
for i in range(m):
        inst = np.random.randint(0,n)

        # Find a nearby positive and negative instances
        pos_list = np.argsort(np.sum((X_pos - X[inst,:])**2, axis=1))
        neg_list = np.argsort(np.sum((X_neg - X[inst,:])**2, axis=1))

        # Select nn nearest hits and misses
        if Y[inst] == 1:
                h = X_pos[pos_list[1:nn+1]]
                q = X_neg[neg_list[0:nn]]
        else:
                h = X_neg[neg_list[1:nn+1]]
                q = X_pos[pos_list[0:nn]]

        denum = np.max(X, axis=0) - np.min(X, axis=0)
        # Update w, note the P(C)/(1-P(class(X[inst]))) = 1 in binary case
        w = w - (1/(m*k))*np.divide(np.sum(np.abs(h-X[inst,:]),axis=0),denum)
```

```
            + (1/(m*k))*np.divide(np.sum(np.abs(q-X[inst,:]), axis=0),denum)
return w
```

2.5. Conclusion

Using a principled design of experiments, we can be fairly certain of these results and the trends encountered therein. Thus we can confidentally say that class-specific correlations provide vital information for the tested machine learning algorithms, and should therefore not be removed during feature selection. Even in the case of a separable feature to rely on, the algorithms performed better under the precense of strong correlations.

These results show that not only do random forests, multilayer perceptrons, support vector machines, and K-nearest neighbors leverage correlations, they do so quite well for strong correlations. Had either F_1 or F_2 been removed through standard feature selection the resulting feature subset would have been entirely inseparable in the non-shifted data, and for the shifted data the accuracy would have been restricted to approximately 60%. This provides good evidence for the power of correlations in machine learning algorithms.

Secondary to these results, rapid increases in computing and data storage capability further the argument against common ideas in feature selection. Machine learning is significantly less restricted by computational complexity than it was during the beginnings of feature selection, and thus there is even less of a reason to restrict ourselves to non-correlated features.

Further, some feature selection algorithms can provide conflicting results in the presence of of correlations. Care should be taken to not inadvertently remove discriminative features be removing correlations which superficially appear to be redundant.

2.5.1. Future Work

As mentioned in the introduction, it is common for black-box explainability methods to break feature correlations when they generate new sample points to test for feature importance. Although this provides results, it is commonly acknowledged that they can be misleading due to the exptrapolative nature of the generated sample points. These points exist in regions not represented by the training data, and thus no guarantees of the accuracy of the machine learning model can be made for those regions. This work's emphasis on the abilities of machine learning models to leverage correlation underscores the need for developments in correlation, or other higher order interaction, to enhance explainability.

3. CORRELATION PRESERVATION SAMPLING

Machine learning (ML) algorithms are utilized in an increasing number of high-consequence application areas; examples include malware detection [103], autonomous vehicles [29], and medical diagnoses [28]. ML models are often a single component in a larger system that must be assessed by a human analyst in order to make decisions with possible high-consequences for false positives or false negatives and can be time sensitive. With this increased usage, the need to understand how a ML model behaves, why a particular output is given, and that the model preserves fairness, is safe, etc. has escalated. As a response, explainability in ML and artificial intelligence (AI) has emerged as a budding research field [4]. However, current explainability methods lack rigor and formality congruent with the consequences that exist in other fields to validate models. In this chapter, we use principles from design of experiments for sensitivity analysis (SA) to define a mathematical framework for *ML explainability* (MLE) methods (*SA Guided Explainability* or SAGE) and focus the discussion on capturing and explaining higher-order interactions in models.

We define MLE as a set of techniques for explaining the inner-workings of a learned model. As many models are sufficiently complex that presenting the inner-working of a model is not decipherable by an end user, many explanation methods for a model or individual predictions present a set of *important features* that contribute to the model output. While there are other approaches to explainability, feature importance is commonly used. We define *feature importance* (FI) as weighting or ranking of the input features based on the impact that a feature has on the output of a model. Most of the approaches assume that input features are independent and are therefore unable to capture any higher-order interactions in the model. Yet, often, the interaction between features is what is the most interesting and discriminative.

The goal of a SA is to verify the consistency of the model behavior or to assess the robustness of simulation results to uncertain inputs or model assumptions. SA can be defined as “the study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input" [98]. In other words, SA seeks to learn relationships between the inputs and outputs of a model. SA uses design of experiments which aims at describing the variation of information under hypothesized varying conditions [77]. There are several SA methods, but we focus here on global, variance-based SA [109] which varies the input and measures how the variations in the input affects variation in the output. SA is used to validate models used in high-consequence applications.

SAGE encompasses three design considerations and highlights gaps in current MLE methods by: (1) managing the source of uncertainty, properly maintaining correlations and input distributions; (2) identifying correct quantities of interest to measure the uncertainty of the output relevant to an explanation; and (3) proportioning the influence of sources of input uncertainty on output

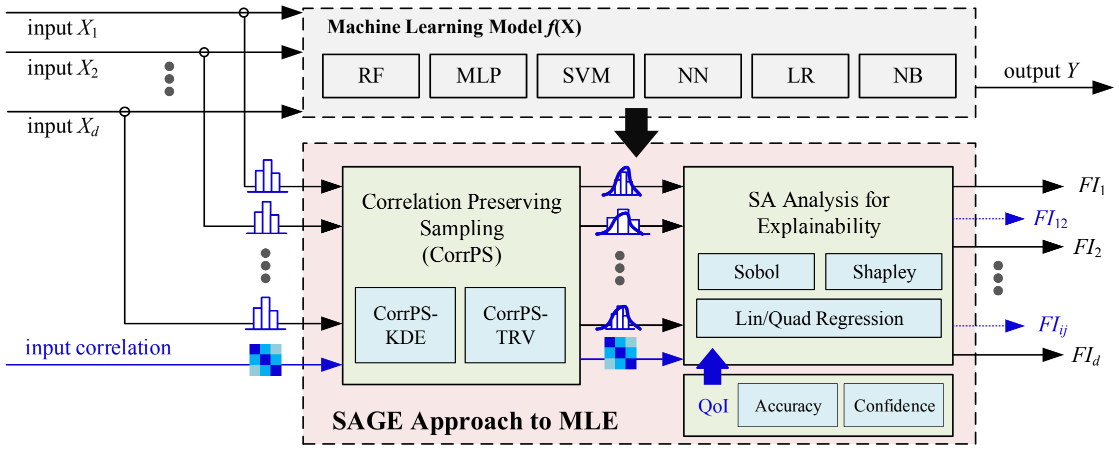


Figure 3-1. SAGE approach to explaining machine learning models. The black arrows represent current steps in MLE. The blue portions highlight the contributions of using SAGE.

uncertainty that accounts for higher-order interactions in the ML model. These considerations are often overlooked. We focus here on the sampling process.

Our contributions include: (1) proposing SAGE as a mathematical framework for improving the rigor in MLE, summarized in Figure 3-1; and (2) proposing two *correlation preservation resampling* (CorrPS) methods that preserve correlations between input features, their input distributions, and produce data samples that are in distribution with the training distribution that can also be used for out-of-distribution detection [63]. We recognize that there is a large body of research that is still lacking and hope that SAGE helps other researchers to frame work to further improve the rigor of MLE.

3.1. Resmapling Approaches

Let $f(\cdot)$ operating on input $\mathbf{X}$ with labels Y such that $Y = f(\mathbf{X})$ represent a black box model where $\mathbf{X}$ is a vector of d input features $\{X_1, X_2, \ldots, X_d\}$. In the context of ML, $f(\cdot)$ represents a learned ML model, however, for SA $f(\cdot)$ can represent any black box model. There are three interdependent principles that affect the results of variance-based SA: 1) the *resampling method* that produces the variance in the input; 2) the *quantity of interest* (QoI) or the output quantity that is measured from $f(\cdot)$; and 3) *how variance in the output proportioned from input variance*. The resampling process needs to have enough variation to produce variations in the output of $f(\cdot)$ which is dependent on the QoI. Further, how the output variance is proportioned to the input variance determines what kind of dependencies are discoverable. If the method assumes all of the features are independent, then higher-interactions in the model will not be discoverable.

Resampling methods refer to techniques that reuse the sampled observations to to reason about a population based on observed sample data and model the population distribution in some form. In ML, typically the problem of sampling is completed (e.g. the use of benchmark datasets). Resampling is used extensively, for example, under the specific techniques of cross-validation and bootstrapping. The most widely used cases of resampling is in class-imbalance where the data is either over- or under-sampled to train models with more balanced class distributions. There are two broad resampling categories: *empirical* and *analytical*.

3.1.1. Empirical Methods

Empirical methods are commonly used by the ML community and refer to techniques that randomly draw values from the observed training set. These methods are simple to implement and often have low computational complexity. However, these methods in general ignore dependence among the features and other relationships among the observed features and examples. Some extensions are used to mitigate these issues such as block bootstrap (blocking certain features or samples to be selected together) or stratified cross-validation to preserve the class distribution.

Bootstrapping is a commonly used method that randomly resamples the observed samples with replacement. Perturbation importance [13] uses feature-level bootstrapping for determining FI at the model level. Bootstrapping ensures that each feature will have realistic values as they can only take on observed values. However, feature-level bootstrapping assumes that all the features are independent and does not preserve correlations between features. This will produce samples that are not realistic, but ensures that valid features are used.

3.1.2. Analytical Methods

Analytical methods create a probabilistic model of the features consistent with the observed training data and can also incorporates constraints of the system being model such as known physical properties. There are two steps to resampling using the analytical methods: 1) train a model on the observed data and 2) draw random samples from the trained model. For these methods, correlations and higher-order relationships between features *can* better be preserved but at the expense of computational complexity. One distinct advantage is that non-observed values can be used rather than being strictly limited to the value present in the observed samples.

There is a rich field of generative Bayesian and deep learning methods []. These methods seek to model the full joint probability rather than just the class likelihood. Generative models can be computationally expensive, require large amounts of data, are difficult to tune, and are domain specific, but they should be able to capture these higher-order interactions between features. Recent successes have been demonstrated in the image [] and text domains []. We focus on the more generic problem of sampling as opposed to a specific domain.

The simplest analytical methods model each feature independently and fit a distribution to each feature based on a quantity such as maximum likelihood, fit a univariate KDE to each feature, or simply use the empirical distribution for each feature. Other analytic approaches account for the dependencies in the data and model the data using techniques such as multivariate KDEs, copulas, or using a translation model where each feature is represented as a mapping of correlated Gaussian variables. Given a model of the population based on the sampled observations, random samples are drawn using techniques including Monte Carlo simulation, Latin hypercube sampling, and stochastic reduced order models.

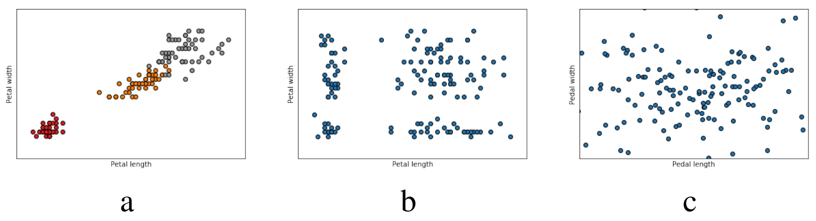


Figure 3-2. The results of sampling on the correlated petal width and petal length features from the iris data set showing a) the original dataset, b) the resampled dataset by perturbing only the petal width feature (bootstrapping), and c) the sampled data following the method used by LIME.

3.1.3. Comparison of Methods

Using the well known iris dataset, we visualize the results from feature-level bootstrapping and a simple analytical method used by Local Interpretable Model-agnostic Explanations (LIME) [93] (a black box MLE method for individual predictions). LIME resamples from a dataset by modeling the training data as a set of independent Gaussian distributions—storing the means and standard deviations of each feature in the training data. Resampled data points are drawn from a normal distribution with the recorded mean and standard deviation. Like bootstrapping, each feature is assumed to be independent and correlations are not maintained.

In the iris dataset, we focus specifically on the third and forth features (petal length and width) which are highly correlated to each other and with the class label. Figure 3-2 shows the original data points (Figure 3-2a) the bootstrapped sampled data points (Figure 3-2b) and the sampling from LIME (Figure 3-2c). Visually, it is clear that 1) the original features are correlated and that 2) the sampled data points are out-of-distribution from the training data and 3) do preserve the correlation between the data points.

3.2. Correlation Preservation Sampling

Recognizing the many sampling methods do not preserve correlations, we propose two *correlation preservation sampling* (CorrPS) methods that preserve correlations between input features, their input distributions, and produce data samples that are in distribution with the training distribution. The first method models the input space using multivariate kernel density estimation to estimate the probability density function of the training data (CorrPS-KDE). The second method inspired by stochastic mechanics uses translation random variables [31] (CorrPS-TRV).

3.2.1. CorrPS-KDE

CorrPS-KDE models the training data using multivariate kernel density estimation [102]. We use Gaussian kernels and use Scott's rule [102] to select the bandwidth of the kernel: $n^{\frac{-1}{d+4}}$. Examples can be obtained by either sampling directly from the multivariate KDE or employing a separate

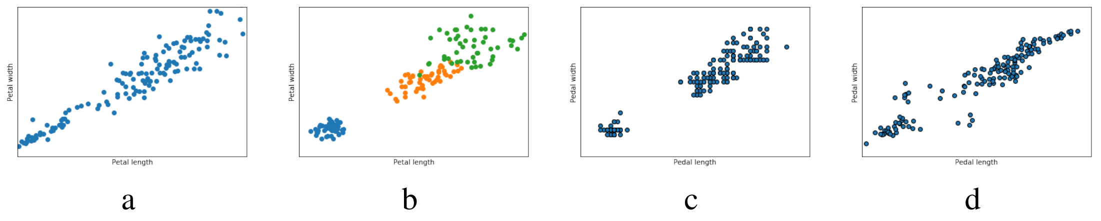


Figure 3-3. The results of sampling on the correlated petal width and petal length features from the iris data set showing a) the CorrPS-KDE, b) the heat map of the PDF from CorrPS-KDE, and c) the sampled data from CorrPS-TRV.

resampling method and using the multivariate KDE to reject examples with low estimated probability densities.

We find that modeling each class with its own multivariate KDE produces better PDF estimates. Figure 3-3a shows resampled data points from a multivariate KDE modeling all of the classes, 3-3b shows resampled data points from a multiple multivariate KDEs fit to each class. The separation between the 'Setosa' class and the other two classes is preserved when using multiple KDEs and is averaged out using only one KDE. This effect could be lessened using different kernels.

The learned KDE could also be used for rejection sampling in conjunction with another resampling technique. For example, candidate examples are chosen using bootstrapping or random sampling between the minimum and maximum values. Examples that with low probability densities according to the KDE are rejected. This approach has the advantage that certain properties of the other resampling techniques are preserved. In Figure 3-3c, a single KDE is used to reject samples from bootstrapped samples. Since bootstrapping will preserve the gap between spaces, the KDE helps preserve the correlations (compare with Figure 4-3b). Also, rejection sampling allows for permutations to be isolated to single variables which is convenient in certain FI methods without loosing correlations.

3.2.2. *CorrPS-TRV*

Recall that $X_1, \ldots, X_d$ represent arbitrary features / inputs to the ML model. We can interpret the available data on these features as samples from some unknown probability distribution. In this section, we propose a model for this distribution that, once properly trained, can be used to create random samples of the X_i that preserve both the marginal distribution of each feature, as well as the correlation among features; these samples can then be used for MLE. The approach is based on the translation random vector model [5, 38].

We model each feature as $X = h(G)$, a function h of a Gaussian random variable G that has zero mean and unit variance. If X is a continuous random variable, then h takes the form

$$h(G) = F^{-1} \circ \Phi(G), \tag{3.1}$$

where Φ is the cdf of G and F is an arbitrary cdf. This particular h is used because

$$\Pr(X \leq x) = \Pr(F^{-1} \circ \Phi(G) \leq x) = \Pr(G \leq \Phi^{-1} \circ F(x)) = F(x)$$

so that X has cdf F. If, instead, X is a discrete random variable that takes values in $\{x_1, \ldots, x_r\}$ with probabilities $p_1, \ldots, p_r$, then h takes a different form, that is,

$$h(G) = \sum_{i=1}^{r} x_i \mathbf{1}(G \in \beta_i), \tag{3.2}$$

where $\beta_1 = (-\infty, a_1]$, $\beta_2 = (a_1, a_2], \ldots, \beta_r = (a_{r-1}, \infty)$ are intervals on the real line, each $a_i = \Phi^{-1}(p_1 + \cdots + p_i)$, and $\mathbf{1}(\cdot)$ is the indicator function, equal to one when its argument is true and equal to zero otherwise. Then, by Eq. (3.2),

$$\Pr(X = x_i) = \Pr(h(G) = x_i) = \Pr(G \in \beta_i) = \Phi(a_i) - \Phi(a_{i-1}) = p_i.$$

We can use this approach to express two features X_i and X_j as $h_i(G_i)$ and $h_j(G_j)$, where G_i and G_j are two standard Gaussian variables with correlation $\rho_{ij} = \mathbb{E}[G_i G_j]$. The correlation between the features is then given by

$$\mathbb{E}[X_i X_j] = \int_{\mathbb{R}^2} h_i(u)\, h_j(v)\, \phi(u, u; \rho_{ij})\, \mathrm{d}u\, \mathrm{d}v \tag{3.3}$$

where

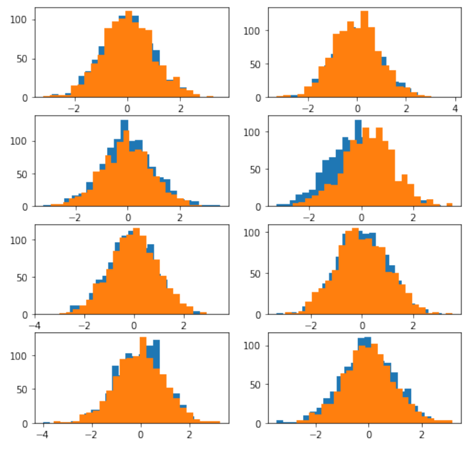
$$\phi(u, v; \rho_{ij}) = \frac{1}{2\pi\sqrt{1 - \rho_{ij}^2}} \exp\left(-\frac{u^2 - 2uv\rho_{ij} + v^2}{2\,(1 - \rho_{ij}^2)}\right)$$

is the joint pdf of G_i and G_j; Eq. (3.3) provides the connection between the correlation of G_i and G_j and the correlation of X_i and X_j.

To create random samples of features $X_1, \ldots, X_d$, we follow four steps. First, given samples of feature X_i, we determine the mapping function h_i. If X_i is a continuous variable, we choose the cdf F using, for example, the empirical cdf or a KDE. If X_i is a discrete variable, we estimate the probabilities $\{p_i\}$ from the data. Second, for each pair of features X_i and X_j, we estimate the correlation between the variables using a standard estimator. We then utilize Eq. (3.3) to determine the value for ρ_{ij} that produces this correlation $\mathbb{E}[X_i X_j]$; this will require an iterative method such as the bisection method. Third, we create random samples of Gaussian vector $(G_1, \ldots, G_d)$ with zero mean and correlation matrix $\{\rho_{ij}, i, j = 1, \ldots, d\}$ where $\rho_{ii} = 1$; there are standard algorithms to do this. Finally, we map each sample of G_i to the corresponding value for the feature as $X_i = h_i(G_i)$.

3.3. Experiments

In this section demonstrate the effects f each aspect in SAGE and their downstream ramifications. Quantifying ground truth explanations or quantifying the goodness of an explanation is an



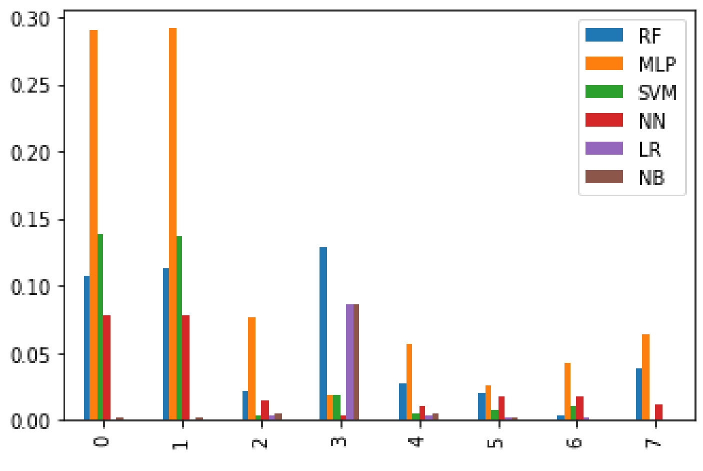


Figure 3-4. a) Histogram for the eight features in the synthetic dataset. There is slight class separation in feature 4 (upper right) and features 1 and 2 are correlated (upper left). b) The change in classification accuracy when removing a feature and retraining.

unanswered research topic. Complicating the issue further, the explanation depends on both the data and the ML model. Therefore, in this work limit our analyses to relatively simple datasets that we can reason about and specifically examine correlation and class separation in the features. Here, we highlight the results on a synthetic dataset composed of eight features that are modeled by Gaussian distributions and class overlap is controlled by adjusting the mean. *All* of the features have the same mean and standard deviation, thus there is no easily separable feature for discrimination except for feature 4 which has slight class separation (the means for each class are off-set) (Figure 3-4a). We set the first two inputs to be correlated with each other. The other features are meant to be noise features and to establish a baseline for irrelevant features.

To establish a notion of ground truth, we remove one feature at a time and retrain the ML model to measure which feature has the greatest impact on the classification accuracy. We also compare the performance of each algorithm without the feature with class separation, thus, we are able to control for higher-order correlations and class separation. In practical problems it is not feasible to retrain the model for removing every individual or control characteristics of the data. Our intention here is to show the dependence of each component in SAGE and how some minor modifications allow for high-order relationships to be discovered. We examined six learning algorithms on the datasets: 1) Random Forest (RF), 2) Multilayer-Perceptron with 5 hidden nodes (MLP), 3) Support Vector Machine with an RBF kernel (SVM), 4) k-NN (NN), 5) Logistic Regression (LR), and 6) Naïve Bayes (NB). The LR and NB algorithms establish a baseline of considering each feature independently.

The results for removing a feature and retraining for both datasets are shown in Figure 3-4b and Table 3-1 shows the accuracies from the examined Ml algorithms. With the exception of LR and NB, all of the algorithms decrease significantly in classification accuracy when either one of the correlated features are removed in both datasets. The feature with class separation causes some interesting results. First, removing the feature with class separation causes a significant *increase*

Table 3-1. Accuracies from the examined learning algorithms on the synthetic datasets.

	RF	MLP	SVM	NN	LR	NB
Correlated	86.6%	88.3%	78.6%	83.5%	52.8%	53.6%
Correlated and Class Separation	70.7%	90.3%	81.8%	84.5%	62.2%	63.0%

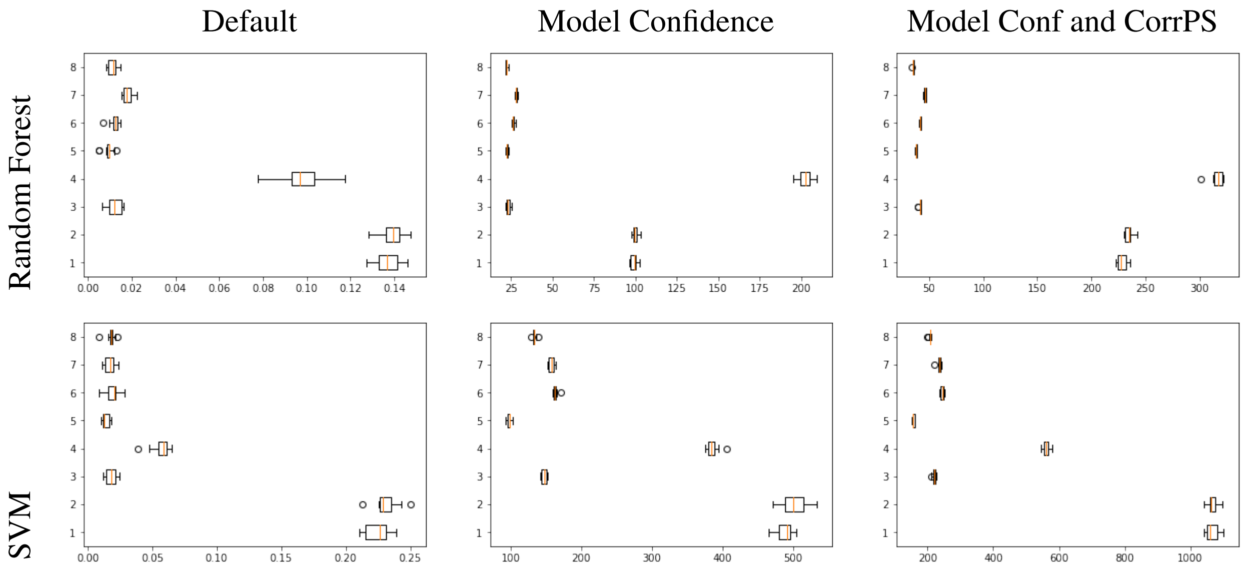


Figure 3-5. A comparison of the FI values using bootstrapping and classification accuracy (Default), Bootstrapping and model confidence (Model Confidence), and CorrPS-KDE and model confidence (Model Conf and CorrPS). Model confidence and CorrPS produce results more consistent with the established ground truth.

in accuracy for the RF. For MLP and SVM, removing feature 3 has moderate impact compared to the correlated features noting that MLP and SVM key on higher-order interactions. We should expect similar results for FI measures.

3.4. Conclusion

Identifying the most influential features for a ML model globally, for each class, and for individual predictions provides transparency and model interpretability that helps ensure model correctness and builds trust. Most model-agnostic methods rely on measuring how variations in the feature space and affect the output of the model. The specific method used for sampling (or equivalently, perturbing) the input feature values has significant ramifications. Most current sampling methods do not preserve correlations between input features, breaking higher order interactions in the model and are inconsistent with the training set data distribution. In this paper, we presented two sampling methods that maintain feature correlations and produce samples that are consistent with the training set distribution. Our methods show more consistent results on simpler datasets that we are able to understand and verify. Additionally, the model used for sampling can be used to provide information about if a data point is out-of-distribution from the training set and should be rejected by a learning algorithm.

4. LIMITATIONS OF CURRENT MACHINE LEARNING EXPLAINABILITY METHODS

Artificial intelligence (AI) and Machine learning (ML) techniques are being used in increasingly more applications including high-consequence applications such as malware detection [103], autonomous vehicles [29], and medical diagnoses [28]. Wide-spread adoption, though, is limited due to a recognized need to trust the models before they are deployed and integrated into larger systems. In response, several explainable AI (xAI) techniques have emerged [4]. However, current xAI methods often lack verifiable foundations and uncertainty quantification—leaving the end user or ML practitioner to decide *if the explanation is valid* and what to do with the provided information. Many xAI methods have justified their approaches by examining how similar they are to how an end-user would explain a decision [93, 125] with accompanying frameworks that define explainability in the context of user-based explanations [27, 59]. While these have laid the foundation, reliance on human evaluation of fidelity may bias explanations towards persuasive explanations rather than accurately describing the learned model [45]. Computational-based explanation fidelity remains an open research question, with most work on xAI fidelity focusing on modified backpropagation and saliency-based methods [2, 81, 105]. Therefore, we propose to define explanation fidelity as a relative accuracy to an analytical solution of feature importance describing how well the explanation describes a model. We examine here the fidelity of post-hoc, model agnostic explainability methods using non-intuitive domains where the meaning is *not* obvious through visual inspection.

We look to the validation and verification (V&V) principles that ensure the correctness of computational modeling and simulation in many science and engineering disciplines [82, 107]. Uncertainty quantification (UQ) and sensitivity analysis (SA) are fundamental elements of most V&V practices and are often applied in tandem. It is our intent to examine global SA (GSA) [98] methods that are well suited for data-driven UQ analysis with the goal of developing credible explanations of an ML model. We examine GSA methods due to their shared objective with xAI of determining the most influential input on the model output and similar implementations.

In certain cases, closed form analytical solutions are possible providing a ground truth—which we exploit in our analysis of explanation fidelity. We find that current xAI methods should only be used in very low risk applications because they fail to capture prediction uncertainty and make several simplifying assumptions that have significant ramifications on the resulting explanations. xAI methods: (1) fail to capture nonlinear interactions in the model and (2) misrepresent the importance of correlated features. The second point, assuming feature independence, has been observed to create incorrect explanations previously [1, 118]. However, demonstrating that nonlinear feature interactions in the model cause incorrect explanations and comparing the two has not been demonstrated in previous literature to the best of our knowledge. This finding is particularly important as most learned models have nonlinear feature interactions.

Our primary contributions include: (1) developing a definition of explanation fidelity relative to a ground truth explanation found by a closed-form analytical solution using GSA, (2) exposing limitations of post-hoc xAI methods using this definition of fidelity—specifically that nonlinear interactions in a model have larger ramifications on the fidelity of an explanations than input correlations, and (3) identifying research gaps that, if addressed, would result in higher-fidelity explanations. We also compare xAI explanations with expert derived solutions demonstrating low fidelity beyond our synthetic dataset. Highlighting these deficiencies helps channel research efforts to address these gaps and improve the credibility of ML models and their usage in high-consequence applications.

The rest of the paper is organized as follows. We first present preliminary background on GSA. Section 4.2 discusses the trust of black box learned models and presents our definition of explanation fidelity. We then use this definition of fidelity to empirically examine the fidelity of several xAI methods in Section 4.3. We provide a discussion of the current gaps of xAI couched in the GSA framework in Section 4.4 before concluding.

4.1. Global Sensitivity Analysis Methods

GSA apportions the influence that input or model parameter uncertainties have on the uncertainty in model output [98]. GSA has a history of application to black box models in the science and engineering disciples and is gaining more traction for systems modelling and policy support [92]. Let $\mathbf{x} \in \mathbb{R}^d$ represent an input vector, $y \in \mathbb{R}$ represent an output (label) and $f : \mathbf{x} \mapsto y$ represent a function that maps $\mathbf{x}$ to y. If we assume that the model parameters do not change, such as inference for an ML model, GSA proportions the uncertainty in each of the d input variables on the output uncertainty measured by a quantity of interest (QoI). We examine two common methods: (1) variance-based Sobol' indices [109] and (2) game theoretic Shapley values [104]. Here, the output GSA is a set of feature importance values $\Phi := \{\phi_j\}_{j=1}^d$ for each variable x_j.

4.1.1. Sobol' Indices

Assuming that $\mathbf{x}$ is composed of d mutually independent random variables, and that the output y is a scalar, the high-dimensional model representation expands a multivariate function $y = f(\mathbf{x})$ as:

$$y = f_0 + \sum_{j=1}^{d} f_j(x_j) + \sum_{k=j+1}^{d} f_{j,k}(x_j, x_k) + \cdots + f_{1,2,\ldots,d}(x_1, x_2, \ldots, x_d) \tag{4.1}$$

where x_j represents the j^{th} input variable, $\mathbb{E}$ denotes expectation, and

$$\begin{aligned} f_0 &= \mathbb{E}[y], \\ f_j(x_j) &= \mathbb{E}[y|x_j] - f_0, \\ f_{j,k}(x_j, x_k) &= \mathbb{E}[y|x_j, x_k] - f_0 - f_j - f_k, \\ &\cdots \end{aligned}$$

Further, using variance to measure uncertainty, and assuming that $f(\mathbf{x})$ is square-integrable and that each variable has finite variance, the variance of y from Equation 4.1 can be decomposed as:

$$Var(y) = \sum_{j=1}^{d} Var_{x_j}(\mathbb{E}_{x_{\sim j}}[y|x_j]) + \left[\sum_{k=j+1}^{d} Var_{x_{jk}}(\mathbb{E}_{x_{\sim jk}}[y|x_j,x_k]) - V_j - V_k\right] + \cdots \tag{4.2}$$

where $V_j := Var_{x_j}(\mathbb{E}_{x_{\sim j}}[y|x_j])$ represents the variance contribution to y from input x_j alone. Thus, Sobol' indices provides a decomposition of the variance in y for each input variable as well as combinations of the input variables. First order Sobol' indices are computed as $S_j = \frac{V_j}{Var(y)}$ and can be extended to higher order indices as $S_{j,k,\ldots} = \frac{V_{j,k,\ldots}}{Var(y)}$. Due to computational overhead of examining all possible sets of features, Sobol' indices are not generally examined beyond the first, second, and total order indices; where total order indices quantify how much inputs or parameters contribute to the total variance on its own and through interactions with other inputs or parameters.

Practically, the sensitivity of y with respect to an input variable x_j is measured by varying the input either by sampling directly from a analytical distribution or from a dataset. Key to GSA are a proper sampling technique and a proper QoI which measures the uncertainty of the model's output. In xAI, this becomes challenging when classification is the QoI as large amount of variances are sometimes needed to change the classification while model confidence measures have been shown to be difficult to calibrate [39], often meaningless [44], and easily manipulated [80].

4.1.2. *Shapley Values*

Recently, a connection was made between the variance-based Sobol indices and Shapley values from cooperative game theory [104]. Let $\mathbb{S}$ be a subset of all input variables $\mathbb{M} := \{x_j\}_{j=1}^{d}$, and $v|_{\mathbb{S}}$ be a value function that approximates the QoI on the given subset of variables. Shapley values proportion a global reward according to individual contributions in a team effort, defined as [124]:

$$\phi_j = \frac{1}{|\mathbb{M}|} \sum_{\mathbb{S}\subseteq\mathbb{M}\setminus\{x_j\}} \frac{|\mathbb{S}|!(|\mathbb{M}|-|\mathbb{S}|)!-1}{|\mathbb{M}|!} \left(v|_{\mathbb{S}\cup\{x_j\}} - v|_{\mathbb{S}}\right), \forall\mathbb{S}\subset\mathbb{M} \tag{4.3}$$

In terms of feature importance, the impact each variable x_j is evaluated over all possible subsets $\mathbb{S}$. It is assumed that f is sufficiently complex that a simpler, surrogate model v is needed for computational efficiency. Shapley values have been used in GSA as a measure of variable importance [83, 66] and are theoretically bound by the first and total order Sobol' indices [52]. Similar to integrated gradients [115], Shapley values calculate the difference between the average and the actual output.

4.2. Trusting Learned Models

The need to trust learned models has been explored broadly in ML [36] and reinforcement learning [119], as well as for specific applications such as computer vision [87, 14] and automotive software engineering [11]. However, most of these studies focus primarily on how robust ML models are to adversarial attacks or out-of-distribution data points. Additionally, with the increased usage of AI in many businesses, several maturity models have been put forward to assess if a learned model is ready to be deployed. Most of these focus on AI operations from a strategic and principled development point of view rather than on an examination of a learned model [99, 72, 3]. For example, guidelines from Ethical AI point out the need to provide explanations and transparency, but do not examine if the explanation reflects the underlying model [116]. As xAI methods have been proposed as a means of providing trust and verifying model behavior, we examine explanation fidelity from the perspective of GSA which have been used in V&V to ensure safety and increase trust in model.

4.2.1. Black Box Explanation Methods

There are several connections between GSA, specifically Sobol' indices and Shapely values, and xAI. LIME [93] was one of the first methods to gain traction in explaining the predictions from an ML model as a means for building trust and examining that a model functions properly. The explanation is derived from a locally weighted linear regression model. To create this linear model, in its simplest form, the input space is randomly sampled and the sampled data points are passed through the model that is to be explained. Each sampled data point and its classification is weighted based on its distance from the data point to be explained. The linear model is then learned using a weighted linear regression algorithm. Explanations are then derived using the input values of the data point to be explained and the weights in the linear model.

SHAP [69] builds on cooperative game theory computing Shapley values (Equation 4.3), and appears to be the strongest theoretically grounded xAI method. SHAP calculates Shapley values using bootstrapping methods to replace a feature with noise to effectively remove it. To deal with high-dimensional data, SHAP is extended to Kernel SHAP where the subsets are weighted based on the weights of their contributions and also has several model specific implementations (e.g. TreeSHAP [68]).

To make computation feasible, most xAI methods, including LIME and SHAP, make the following limiting assumptions: (1) mutual independence of input variables—ignoring all dependencies; and (2) a linear relationship between the output QoI and all input features—enabling the computation of expectations directly rather than from sampling [69]. Follow up work has proposed extending SHAP to model input dependencies (SHAP-Dep) [1] incurring an increase in computational complexity.

We denote the importance values estimated using xAI methods as $\hat{\Phi}i := \{\hat{\phi}_j\}_{j=1}^{d}$ estimated by an explainability value function v. For consistency in notation, we denote the dependence of v and f on both $\mathbf{x}$ and the restriction to the subset of features $\mathbb{S}$ with the expressions $v|_{\mathbb{S}}(\mathbf{x})$ and $f|_{\mathbb{S}}(\mathbf{x})$. The validity of the explanation depends on how well $v|_{\mathbb{S}}$ approximates $f|_{\mathbb{S}}$ for all $\mathbb{S} \subseteq \mathbb{M}$. In practice, having a closed-form solution to Φ is not available or is restricted due to computational constraints, therefore $\hat{\Phi}$ is a numerical approximation of the analytical solution, Φ.

4.2.2. What Constitutes the Fidelity of an Explanation?

There are several measures that can be used to evaluate an explanation [75]. Here, we study *fidelity* which we define as the accuracy of the explanation to the underlying model. An explanation that has complete fidelity would describe the model in detail. Therefore, to be useful to an end-user a trade-off exists between fidelity and completeness of an explanation to convey enough information to describe the decision process. As noted earlier, reliance on human evaluation of fidelity may bias explanations towards persuasive explanations rather than accurately describing the learned model [45]. Here, we suggest definitions for an explanation and a mathematical notion of *fidelity* based on Shapley values.

Definition 4.2.1. An *explanation* is a subset $\mathbb{S}^\star$ of $n \leq d$ features with corresponding importance values $\Phi_{\mathbb{S}^\star} := \{\phi_j\}_{x_j \in \mathbb{S}^\star}$ from the set of all features $\mathbb{M}$ that have the greatest contributions to $f(\mathbf{x})$:

$$\mathbb{S}^\star = \underset{\mathbb{S} \subseteq \mathbb{M}, |\mathbb{S}|=n}{\arg\max} \sum_{x_j \in \mathbb{S}} \phi_j \tag{4.4}$$

where the ϕ_j correspond to 4.3, such that the nominal feature values are defined by the realization of a data point $\mathbf{x}$.

Definition 4.2.2. The *fidelity* $\mathbb{F}_j$ *of an explanation,* $\mathbb{S}^\star$*, of* $\mathbf{x}$ *for the* j^{th} *feature* is the complement to the absolute difference between the actual ϕ_j values and the estimated $\hat{\phi}_j$ values:

$$\mathbb{F}_j(\mathbf{x}, v) = 1 - |\phi_j - \hat{\phi}_j|. \tag{4.5}$$

This definition assumes that the ϕ_j values are in the range of [0,1] which can easily be done by normalizing the values Φ. We subtract this quantity from one so that $\mathbb{F}_j$ near zero/one correspond to low/high fidelity. An aggregate score can also be obtained by summing the individual $\mathbb{F}_j$ scores, while care should be taken as the score could vary based on the number of features considered. This definition is defined with respect to a feature as well as a particular data point. The fidelity of an explanation may vary per feature and in different areas of the input space. Therefore, the fidelity $\mathbb{F}_j$ is dependent on how well $v|_{\mathbb{S}}(\mathbf{x})$ approximates $f|_S(\mathbf{x})$. Equation 4.5 is equivalently expressed as:

$$\mathbb{F}_j(\mathbf{x}, v, f) = 1 - |v|_{\mathbb{S}}(\mathbf{x}) - f|_S(\mathbf{x})| \tag{4.6}$$

As defined in Equation 4.4, an explanation is a ranked feature list, therefore, v does not necessarily have to be equivalent to f to produce useful explanations, but it should be close enough to produce the same ordering and give an idea of the magnitude of each feature's importance. Therefore, while absolute difference can serve as an appropriate loss function, a metric such as the Kendall Tau metric could also measure the difference in the feature rankings. Practically, calculating S^* would require relearning $f|_S$ for all $\mathbb{S} \subseteq \mathbb{M}$ making it computationally infeasible for all but the most trivial problems, hence the need for and importance of v to accurately and efficiently approximate f. It should be understood how v differs from f and what uncertainty comes from the model-form error.

4.3. Empirical Examination of Explanation Fidelity

This section examines *the fidelity of explanations* for models with various properties. To establish ground truth explanations, we examine: (1) using closed-form analytical solution to Shapley values (Equation 4.3) on a synthetic model where we can manipulate the properties of the features and the model and (2) using an ensemble of expert explanations in a subjective domain (cybersecurity). In this case, cybersecurity experts are required to not only make a diagnosis of maliciousness but also provide indicators that support their claims. Thus, the explanations are part of the triage process and a natural fit for xAI methods. We compare the global measures of LIME [93], SHAP [69], and SHAP-Dep [1] aggregating over the entire training set with: (1) permutation feature importance [13] on a random forest trained on data generated by the model using sklearn[1], (2) the GINI values from the same random forest model, (3-4) empirical and analytical Sobol indices (implemented in OpenTurns[2]), and (5-6) empirical Shapley values.

4.3.1. Synthetic Data

We use simple regression models with four input variables, with and without correlation, and with and without non-linear feature interactions:

$$y = 2x_1 + 3x_2 + x_3 + x_4 \tag{4.7}$$

$$y = 2x_1 + 3x_2 + x_3 x_4 \tag{4.8}$$

$$x_1 \sim \mathcal{N}(0,1), x_2 \sim \mathcal{N}(0,2), x_3 \sim \mathcal{N}(0,3), x_4 \sim \mathcal{N}(0,4) \tag{4.9}$$

For correlation, we set x_1 and x_2 to be perfectly correlated as a worst-case scenario. We build on the traditional notion that greater weights and variance equate to a larger importance factor (in this case normalizing the data would negate this, but for demonstrative purposes we keep the different variances and variance is important for the GSA methods). Given the actual models and low number of features, we can calculate the Shapley values considering *all* of the feature combinations (see Appendix B for details). The explanation fidelity for each feature (Equation 4.5) is computed using the Shapley value ϕ_j and the value from each xAI method as $\hat{\phi}_j$.

We create a dataset from each model with and without correlated features by sampling 1000 samples for each x_j and recording the resulting y from Equations 4.7 and 4.8. We then train a random regression forest on the data as our black box model (we tested several ML algorithms; the most important factor was the ability to model higher-order interactions, so models that make strong independence assumptions such as naïve Bayes had significantly different results). The fidelity of the explanations showing the differences between the Shapley values and the calculated importance for each feature are shown in Figure 4-1a-d for each combination of input independence/correlation and model linearity/non-linearity. It is clear that while correlations have an impact on the fidelity, non-linear feature interactions produce even lower fidelity values. We make the following observations:

[1] https://scikit-learn.org/

[2] https://pypi.org/project/openturns/

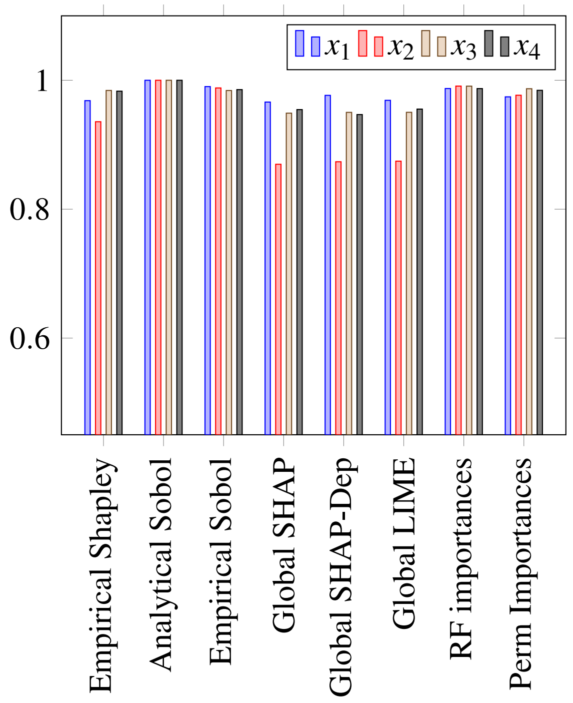
1. When the model meets the correct assumptions of feature independence and linearity (Figure 4-1a), the explanations best match the ground truth, although x_2 is consistently underestimated by LIME, SHAP and SHAP-Dep.
2. When only correlation is introduced (Figure 4-1b), the fidelity of each method degrades as expected.
3. Once non-linear feature interactions are introduced (Figures 4-1c and d), the fidelity significantly decreases compared with just correlation. The features with a non-linear interaction are under-valued and the features with a linear interaction are over-valued. Black box xAI and Sobol' methods do exceptionally poor in capturing non-linear interactions in the model—LIME performing noticeably worse than SHAP and SHAP-Dep.
4. While SHAP-Dep is designed to work specifically with correlated features, we find that it shows lower model fidelity than SHAP in the presence of non-linearity—this is probably partially due to the fact that we used treeSHAP as the underlying version of SHAP which is not available in SHAP-Dep. Additionally, the sampling complexity requirements are larger to model dependencies as demonstrated in the following section.
5. The tree-based importance methods consistently show high model fidelity and do exceptionally well in the case of correlated input features and non-linear model interactions.

With these results, we strongly caution the use of SHAP, SHAP-Dep, and LIME when the ML models include correlated features or non-linear interactions. Of course, it should be noted that SHAP, SHAP-Dep and LIME are are designed for local explanations for a specific data point. While the importance features from the random forest methods show the highest model fidelity, they only provide global feature importance—but do show robustness and high fidelity explanations.

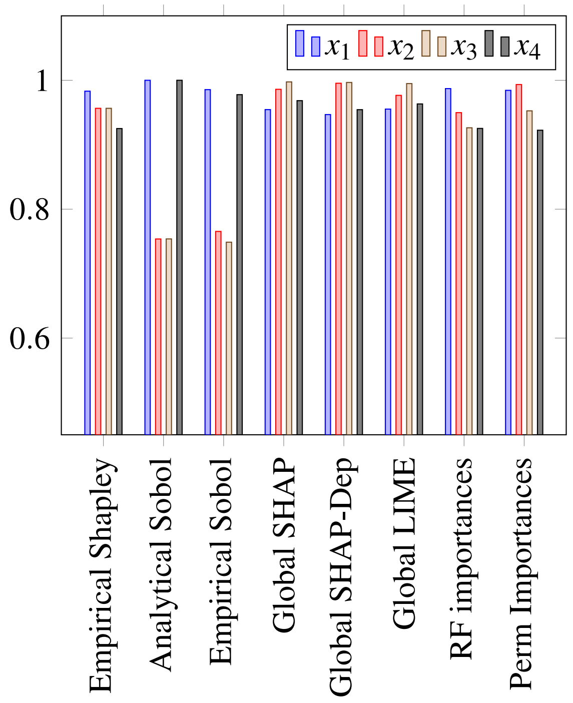
4.3.2. Ensemble of experts comparison

We examined the explanations from a random forest consisting of 500 trees trained on data from PDFrate [108] to detect malicious PDFs. Considering that SHAP performed better than LIME, we examined SHAP-Dep and TreeSHAP. We compare the generated explanations with an ensemble of expert explanations from five cybersecurity experts who analyzed a PDF from PDFrate and provided the features for *why* they classified it either as malicious or benign—which is consistent with their daily tasks in determining maliciousness *and* why by pointing out specific indicators. We compared the xAI explanations with expert explanations on 20 samples (more details can be found in Appendix D). The class labels for malicious PDFs are often difficult to obtain and the features are wide ranging such that it is often easier to detect maliciousness rather than the lack of maliciousness. It is also a very non-intuitive domain—visual inspection of a PDF does not immediately determine what the classification should be.

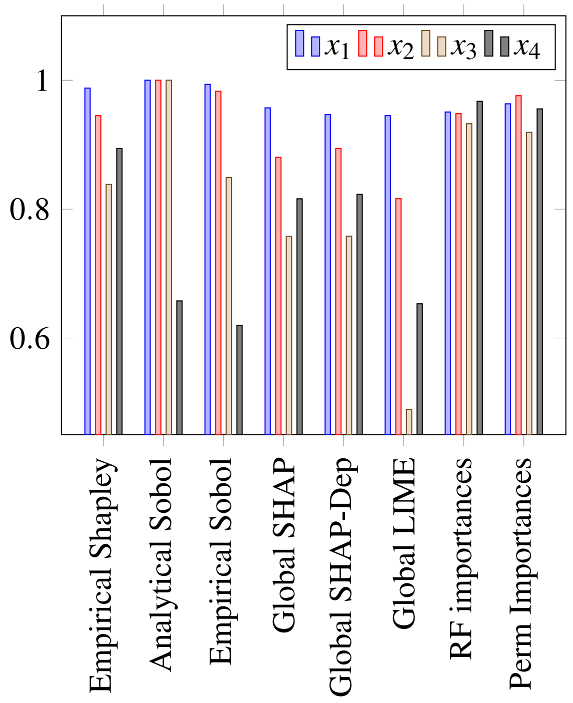
The explanations for the predictions of four PDFs are shown in Table 4-1. The weights for the "Expert" are the number of experts who used a particular feature as an indicator for maliciousness or being benign. The first three examples are confidently classified as malicious PDFs by cyber experts. The last PDF represents one that looks to be benign but with lower confidence. The first



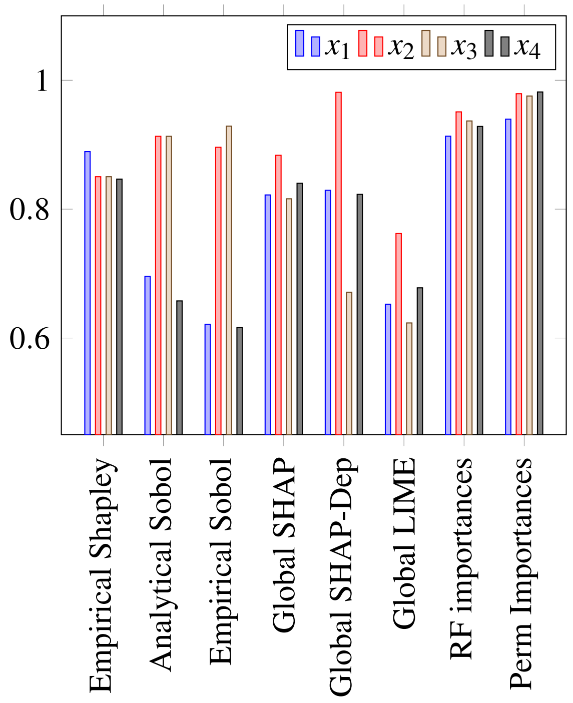
a) Independent Features; Linear Model



b) Correlated Features; Linear Model



c) Independent Features; Non-linear Model



d) Correlated Features; Non-linear Model

Figure 4-1. The fidelity of model explanations per feature. Here, a value of 1 is perfect fidelity. Non-linearity in the model has the greatest negative impact on fidelity

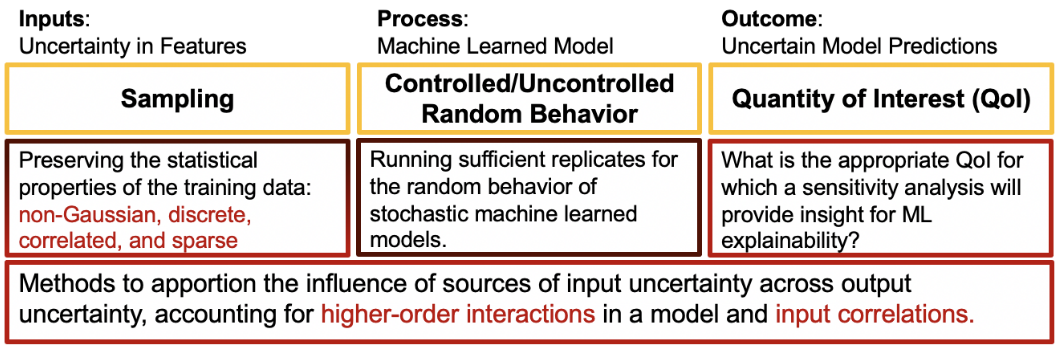

Inputs:
Uncertainty in Features
Process:
Machine Learned Model
Outcome:
Uncertain Model Predictions
Sampling
Controlled/Uncontrolled Random Behavior
Quantity of Interest (QoI)
Preserving the statistical properties of the training data: non-Gaussian, discrete, correlated, and sparse
Running sufficient replicates for the random behavior of stochastic machine learned models.
What is the appropriate QoI for which a sensitivity analysis will provide insight for ML explainability?
Methods to apportion the influence of sources of input uncertainty across output uncertainty, accounting for higher-order interactions in a model and input correlations.

Figure 4-2. Mapping of GSA processes to xAI and highlighting current holes.

five features for each PDF are indicators or explanations for a malicious classification. The bottom five are features that indicate a benign classification with their associated weights. The values in bold correspond to the features that are consistent with the expert explanations. The experts were very consistent in their explanations, very frequently providing overlapping attributes in their individual explanations for a given observation. About half of the experts identified attributes were more abstract than can be captured by a single conventional numeric feature and about half were directly represented by a feature in the machine learning model. We found that overall the explanations from TreeSHAP are better aligned with expert explanations, although often only highlighting the presence of JavaScript. Other explanations did not match the expert explanation in any attribute.

SHAP-Dep often provided irrelevant features as explanations. SHAP-Dep models the feature correlations whereas SHAP treats all features independently and can sample each feature separately. Because of this, SHAP-Dep requires significantly *more* samples to cover a more complex space and to reduce the statistical error from sampling. In the original paper for SHAP-Dep [1], only up to 10 features were considered and this problem was not noted—PDFrate has 135 features.

4.4. Discussion

The overall process of GSA is mapped to the ML paradigm in Figure 4-2 and composes four major components: (1) how to sample the data to represent uncertainty in the inputs, (2) running sufficient replicates or trials to understand the behavior of the model (in GSA, this is often a complex model that is expensive to execute—different from the inference stage in most ML models), (3) measuring an appropriate QoI—particularly for explainability purposes and capturing appropriate uncertainties, and (4) methods to apportion the source of input uncertainty to the output uncertainty. Examining xAI techniques through the lens of GSA, we have identified three primary areas where improvement could significantly improve the fidelity of the explanations:

Sampling Most xAI and GSA methods assume that the inputs are independent. As has been previously discussed here and in other works [118], independence assumptions about the input

Table 4-1. Examples explanations on prediction of malicious PDFs by an ensemble of experts and SHAP-Dep and TreeSHAP.

	Expert		SHAP-Dep		TreeSHAP	
HASH: 0c8f17b213	**cnt_javascript**	5	cnt_page	0.27	**cnt_javascript**	0.16
	cnt_Encrypt	1	cnt_action	0.23	**cnt_js**	0.05
	–	–	pos_page_avg	0.23	creator_oth	0.04
	–	–	pos_page_max	0.23	producer_oth	0.02
	–	–	cnttrailer	0.19	author_oth	0.01
	–	–	cnt_image_total	0.12	pos_image_max	0.01
	cnt_Encrypt	1	cnt_endstream	0.12	cnt_endstream	0.02
	page_size_Letter	1	title_oth	0.13	pdfid1_len	0.02
	cnt_Metadata	1	producer_oth	0.13	pdfid1_num	0.02
	cnt_obj	3	cnt_font	0.14	cnt_font	0.050
HASH: 3c0739dd20	**value_timezone**	5	cnt_page	0.22	**cnt_javascript**	0.09
	cnt_javascript	5	cnt_box_other	0.20	**cnt_js**	0.08
	Creator_WPS_Office	3	pos_box_max	0.19	**createdate_tz**	0.05
	cnt_0x0_boxes	3	pos_box_avg	0.18	**moddate_tz**	0.04
	Normal_Metadata	2	pos_page_avg	0.18	**producer_lc**	0.02
	–	–	pdfid1_len	0.14	cnt_eof	0.01
	–	–	pdfid0_len	0.14	cnt_startxref	0.01
	cnt_Metadata	1	producer_oth	0.15	pdfid1_num	0.01
	cnt_obj	1	producer_mismatch	0.15	creator_len	0.02
	value_file_size	1	pdfid1_num	0.16	cnt_font	0.05
HASH: be02394779	**cnt_javascript**	5	moddate_tz	0.18	cnt_font	0.07
	cnt_javascript_obs	4	**cnt_js**	0.15	pos_eof_max	0.04
	cnt_FlateDecode_obs	4	**cnt_javascript**	0.13	cnt_obj	0.03
	cnt_OpenAction_obs	4	createdate_tz	0.13	pos_eof_avg	0.03
	cnt_OpenAction	4	title_mismatch	0.13	cnt_box_other	0.03
	–	–	cnt_action	0.13	–	–
	–	–	ratio_size_page	0.14	–	–
	cnt_obj	1	cnt_js	0.18	–	–
	value_file_size	1	cnt_javascript	0.18	cnt_javascript	0.02
HASH: dfa717d485	cnt_Autoaction	3	pos_page_avg	0.17	–	–
	cnt_Acroform	2	pos_page_max	0.17	–	–
	cnt_JBIG2Decode	1	cnt_page	0.14	–	–
	cnt_FlateDecode	1	createdate_mismatch	0.10	–	–
	cnt_startxref	1	createdate_ts	0.10	–	–
	Normal_Metadata	2	pdfid0_len	0.14	**cnt_obj**	0.02
	Producer_Adobe	2	cnt_stream	0.15	pos_box_max	0.03
	value_timezone	2	cnt_startxref	0.15	**cnt_js**	0.04
	cnt_Javascript	2	cnt_eof	0.15	cnt_font	0.05
	cnt_obj	3	cnt_endstream	0.15	**cnt_javascript**	0.06

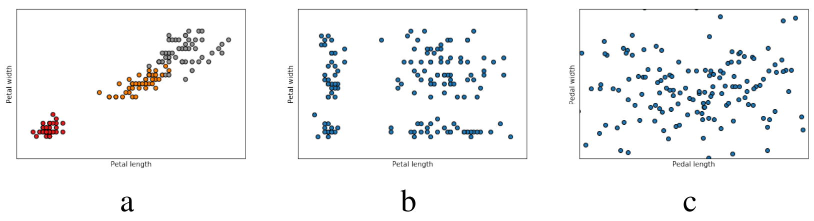


Figure 4-3. The results of sampling on the correlated petal width and petal length features from the iris data set showing a) the original dataset, b) the resampled dataset by perturbing only the petal width feature (bootstrapping), and c) the randomly sampled data.

cause incorrect feature importance values when correlations are present. The appeal of such a limiting assumption is motivated by avoiding the computational cost in modeling the full-joint probability distribution. There also exists a plethora of readily available expedient sampling techniques such as bootstrapping and independent random sampling that are commonly used. Although, proper care is rarely taking into consideration with regards to the unrealizable data points they create data which are clearly out of distribution. For example, consider the third and forth features (petal length and width) of the Iris dataset which are highly correlated to each other and with the class label. Figure 4-3 shows the original data points (Figure 4-3a) the bootstrapped sampled data points (Figure 4-3b) and random sampling (Figure 4-3c). Visually, it is clear that: (1) the original features are correlated, (2) the sampled data do not preserve the correlation between the data points and (3) therefore, the sampled data points are out-of-distribution from the training data. What does an explanation generated from these types of data points tell us about the model?

Quantity of Interest For xAI, the output of the classifier or a confidence metric is often used, but is that really what is most important for explaining a prediction? As pointed out by Rudin [97], often just knowing *where* a model "looks" is not always sufficient for *why* a prediction was made. Future work should investigate other possibilities in QoIs that correlate with explanations.

Non-linear Uncertainty Apportionment We have found that the lack of ability to apportion the influences of input uncertainty across output uncertainty accounting for higher-order, nonlinear interaction in a model is the greatest challenge. This poses a significant hurdle to overcome as most state-of-the-art ML models are highly non-linear and have high-order feature interactions within the model.

These issues are not isolated to xAI, but are also challenges in the GSA and V&V communities. There are several promising research lines that could be leveraged including work from Sobol' indices that specifically address correlations [42, 70] in certain situations and may be applicable for non-linear interactions. To our knowledge, addressing non-linear uncertainty apportionment and appropriate QoIs for explainaiblity have not been explored in much depth. See Razavi et al. [92] for a good overview of current research directions in GSA include perspectives from the GSA community on its use in ML. Appropriate QoIs could leverage recent advances in model output calibration [78] or leverage some other metric which provides insights into the actual decision making process of the learned model, perhaps along the same lines as anchors [94]. Our primary goal is that by exposing these issues they can be addressed more directly to improve the credibility of learned models enabling their confident use in high-consequence applications.

4.5. Conclusion

While xAI has helped to provide insights into the learned models and offers some means of trust, we have shown that strong assumptions that have been made for computational feasibility need to be considered when using xAI—especially in high-consequence applications and in models with nonlinear feature interactions. By casting xAI within the framework of GSA, we identified several holes and demonstrated that correlated input features and, more significantly, non-linear model interactions have strong ramifications on the fidelity of xAI methods. Random forest models are surprisingly robust to these and may offer some paths for future work. New explainability methods are needed that are able to capture higher-order model interactions and can provide estimates of uncertainty. We only examined a small set of xAI methods, future work could expand this notion given the ability to calculate a ground truth explanation value. This is difficult moving into image spaces where the definition of features is less-well defined due to the deep structure of neural networks. Our motivation is that this analysis motivate others to pursue solutions to the gaps in xAI.

5. CLASSIFICATION TRUSTWORTHINESS

As Artificial Intelligence (AI) and Machine Learning (ML) become more ubiquitous in high consequence applications, corresponding laws and regulation are established that mandate explanations for the decisions made by learned models [120]. Therefore, many interpretablity and explainability methods have been proposed [4]. Most of the approaches have assumed that *trust will increase by explaining the prediction*, e.g., through identifying the most important features, and counter-examples. Instead, we examine *how much* we should trust the output of a learned model based on the geometry of the data and the geometry of the data's clusterings and classifications. ML models are trained under the assumption that training and testing data are independent and identically distributed. However, a learned model will make a prediction regardless of whether the test point is drawn from the training distribution. This causes at least four problems: (1) sample-selection bias where the sampled training data is not representative of the real-world data [130], (2) concept drift as the data distribution changes over time [67], (3) introduction of novel classes [71], and (4) adversarial attacks that exploit these vulnerabilities [85].

The concept of *model confidences* seeks to address these problems but has limitations. Out-Of-Distribution (OOD) detection techniques have shown that some model confidences, e.g. *softmax,* are overly confident and in many cases they are uninformative in determining if a data point is out of distribution [44]. Adversarial attacks, by simple modifications to a data point, can easily fool a learned classifier into confidently predicting the wrong class [85]. Robustness defenses work against some types of attacks, but are easily defeated by others [19].

We examine metrics, based on the *geometric relationship* of the training data and the classification boundary rather than *model confidences*. Our geometrically-inspired metrics allow for visualizations that are intuitive and we demonstrate their effectiveness in identifying evasion attacks in detecting malicious PDFs.

5.1. Geometric Trustworthiness

We propose the following set of geometric trust metrics for a test point *P*. Each trust metic provides a different aspect of the trustworthiness of a prediction.

1. **Training Proximity Metric**. Model output is trustworthy if *P* is close to a high density region of training points. See Section 5.1.1.
2. **Extrapolation Metric**. If *P* is within the space of training points, regardless of class, the classification is an interpolation, which can be trusted more than an extrapolation. See Section 5.1.2.

3. **Class Ambiguity Metric**. Classification is less certain the closer *P* is to a classifier decision boundary. See Section 5.1.3.

5.1.1. Training Proximity Metric

We consider the prediction for a point *P* to be trustworthy if *P* is near a cluster of training points with the same class as the prediction for *P*. Figure 5-1a illustrates several possible cases. We employ the HDBSCAN [18] clustering algorithm in our prototype implementation, although any clustering algorithm could be used. HDBScan provides a measure of strengths of membership of a point belonging to its closest cluster (if any is sufficiently close) as well as if it is an outlier which we use as a measure of trust—stronger membership equates to higher trust. Default HDBSCAN parameters were employed, except a minimum cluster size of 10 was employed. We assume that smaller groups of points should be treated as outliers, and not especially trustworthy. The HDBSCAN measure of strength is a value in the range [0,1], where 0 means the point is not part of any cluster, and 1 means it belongs in the cluster.

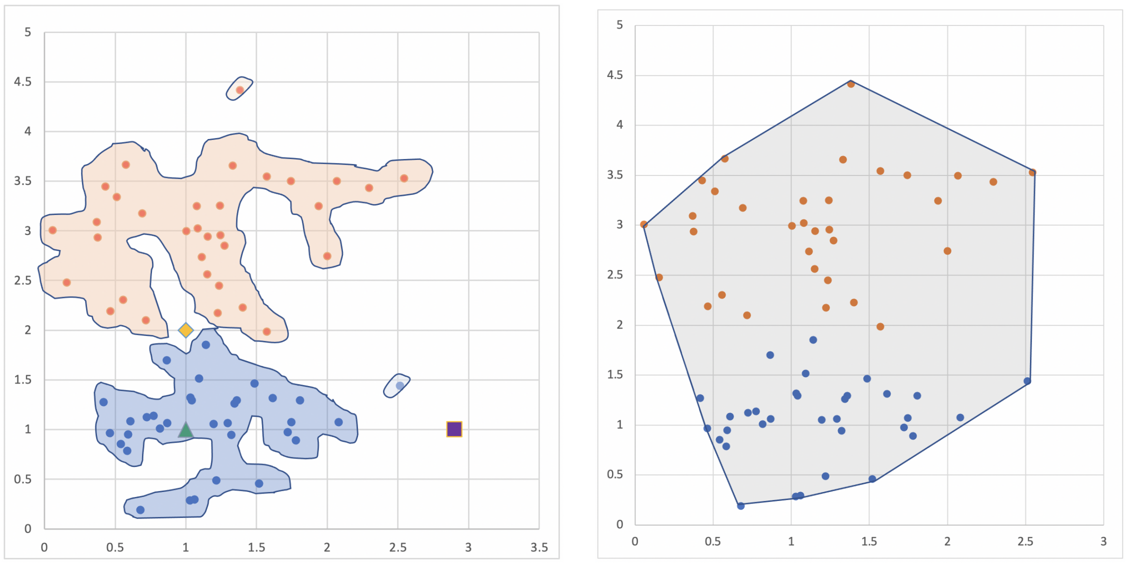


Figure 5-1. a) Training Proximity Metric. Blue and orange training points are from different classes; shaded regions denote their clusters. The green triangle is well within the blue neighborhood; if the classifier assigns it blue, it is highly trustworthy. The classification of the yellow diamond has less trust as it is near both clusters. The purple square is not near either cluster, so its classification is also not trustworthy. b) Extrapolation Metric. Points outside the region are extrapolated, are considered less trustworthy. A point inside the hull is an interpolation of multiple points, so its classification is regarded as more trustworthy, regardless of it being assigned blue or orange. (Best viewed in color)

5.1.2. Extrapolation Metric

We deem that classification from interpolation is more trustworthy than extrapolation. If the classification is extrapolated, the user is trusting that the decision boundary is correctly extended beyond the immediate influence of the training set. We distinguish interpolation from extrapolation by whether the point lies in the convex hull of the training points. Our metric is an approximation of the signed distance from the point to the *convex hull* boundary illustrated in Figure 5-1b. By *signed distance*, we mean that points inside the hull report a negative distance. Interpolated point values are in $[-0.5, 0]$, and extrapolated point values are in $(0, \infty)$, because of the normalization of the training data to the unit hypercube.

Computing convex hulls in high dimensions is computationally intractable [8]. Fortunately, an approximate distance is sufficient. Testing a point P against an n-dimensional convex hull can be approximated by testing P against the hulls of multiple projections to lower dimensional spaces [21]. For each projection, we compute the signed distance d from the projected test point to the 3D hull of the projected training points. We report the maximum d over all projections.

If the point is outside the hull of any projection, the point is outside the n-dimensional hull. In general, the signed distance in a projection is a lower bound on the true distance from the test point to the n-dimensional hull. It is possible that the true distance is larger, and we might misclassify outside points as inside, but the expected inaccuracy decreases the more projections we use. We found multiple 3D projections is efficient and accurate; see Figure 5-2. Here, we do not examine all possible combinations of features, but consider the 1^{st}-3^{rd} features, then the 2^{nd}-4^{th} features until the n^{th}, 1^{st} and 2^{nd} features are considered. The total calculation time is linear in n. In this example, all "in" data points are correctly identified.

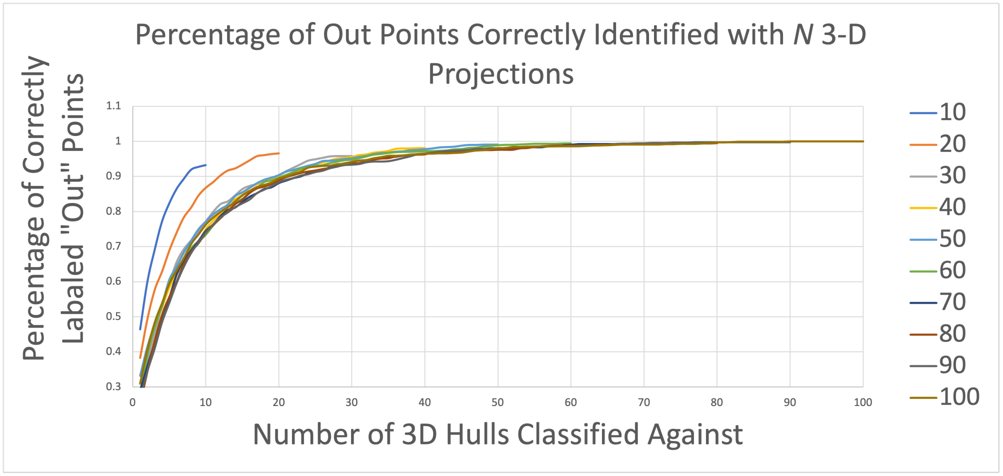


Figure 5-2. Extrapolation Metric categorization accuracy: inside ($d \leq 0$) vs. outside ($d > 0$). Training points are randomly generated in a unit hypersphere, and test points in the enclosing unit hypercube. For each curve we use n 3D convex hulls. We achieve 95 percent accuracy for the 20 dimensional case, and near 100 percent accuracy for 60+ dimensions.

5.1.3. *Class Ambiguity Metric*

Intuitively, the area around a decision boundary exhibits higher uncertainty due to issues such as underspecification [24]. As such, small changes in the feature values of the point can change its classification. Given a point to be classified, P, and it classification C, we start by finding the closest training point TP whose class is not C. A binary search between TP and P locates the decision point, DP, where the classification changes from C to not C.

Inspired from foundational path planning in robotics that exploration of the space leads to better solutions than direct optimization [61], we search near DP for a closer decision boundary point. We choose a vector V with a random direction, and magnitude at most the distance from DP to P. We add V to DP and then scale to ensure that the distance is at most the distance between DP and P to produce a random point RP. If RP's class is C, repeat until it is not C. We repeat searching until the search budget is exhausted. The closest DP found during the search defines the approximate decision boundary distance as illustrated in Figure 5-3.

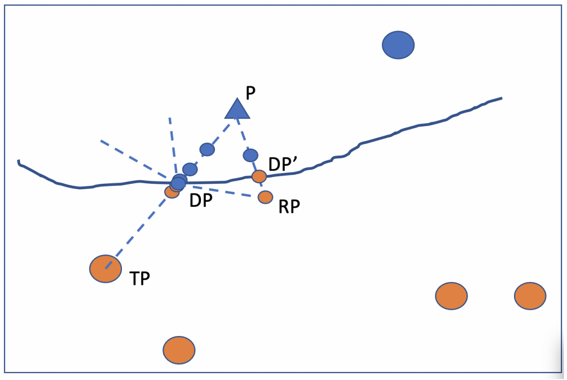


Figure 5-3. Random line search for approximating the distance to the decision boundary. Given test point P (blue), the closest training point of a different classification is TP (orange). A line search between TP and P locates DP, which is approximately on the decision boundary. Going in random directions from DP locates RP with a different class than P. A line search from RP locates DP', which is closer to P than DP. This is repeated. In practice we find a point on the decision boundary that is reasonably close to P.

5.1.4. *Measuring the Trust of a Region*

The above metrics for measuring the trust of the classification of a single point can be repeated to measure trust over a region. There are many possible regions of interest and ways to define them. One could consider the hypercube of the training points, or some subset or superset. If the test points come from a known distribution, that could be used. Sample points can be randomly located within a region, or on its boundary. The average, min, max, and other statistics of the samples' trust would define the trust over the region, with the understanding that this is an approximation limited by discrete sampling.

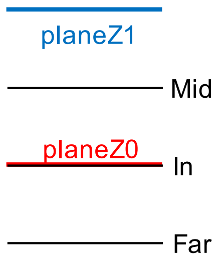
5.2. Identifying Important Features

We provide a geometry-inspired measure of the importance of features (dimensions) depending on point locations. The algorithm computes a vector from a point to the nearest point on the decision boundary, as in the Class Ambiguity Metric in Section 5.1.3. It ranks features according to their relative magnitudes, with shortest vectors being most important.

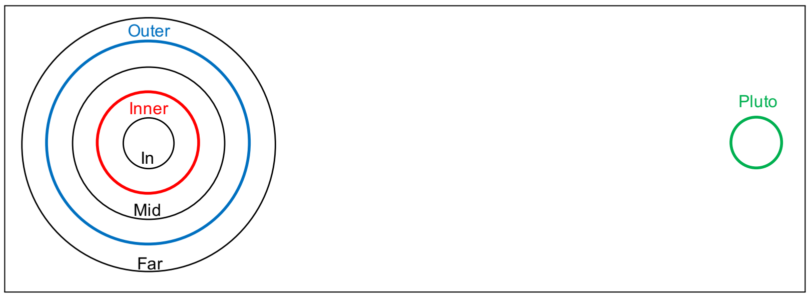
This metric is well-defined for data consisting solely of continuum features, or solely of binary and categorical features. Mixing continuum and non-continuum features require further study, because it is unclear how to scale them for comparison. In particular, $|\mathrm{P}_j - \mathrm{TP}_j| = 1$ for a non-continuum feature j. We experimented with ranking features using the Training Proximity Metric and Extrapolation Metric, but the results were not useful.

5.3. Synthetic Data Verification of the Trust Metrics

For verification of the metrics, we consider various synthetic data sets. The first type of data are from geometric shapes, hyperboxes and hyperspheres. The second type are normal distributions, with varying overlap. For every type we consider both 3D and 8D datasets, i.e. 3 features and 8 features. The training data consists of two classes. For hyperspheres, we also consider the effect of including a third class *Pluto* far from the data of interest, to measure the effects of spurious data. We test using data that consists of (1) *Mid* points midway between the distributions of the two training classes; (2) *In* points near/in one of the classes within the training space; (3) *Far* points on the far side of a class outside the training space; and (4–5) points *A* and *B* from the same distributions as the classes in the training data. See Figures 5-4 and 5-5 and refer to them in the following discussions about the experimental observations on the datasets. Specific details about the training and test datasets are provided in Tables 5-1, 5-2, and 5-3.



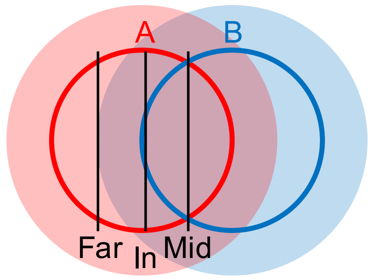
(a) Planes



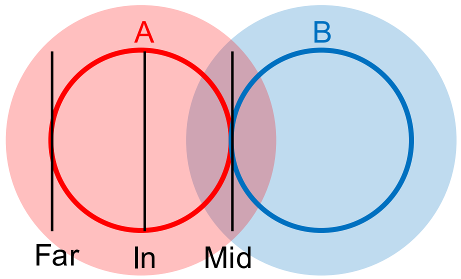
(b) *Rings* & Pluto

Figure 5-4. Synthetic closed-geometry training (red and blue) and test (black) sets. 2D drawing of nD data, to scale.

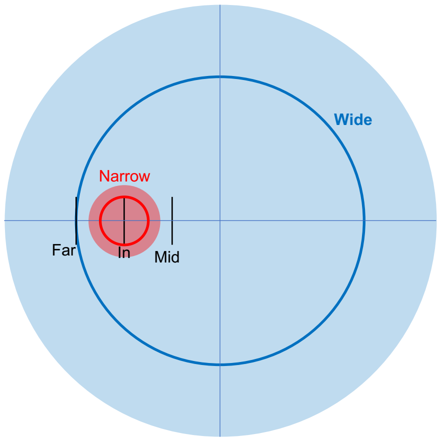
Training classes nominally contain 1,000 points, depending on the dimension and type. Test sets are all 100 points. Points on the hypersphere and hyperplane are selected uniformly at random. The trust metrics were calculated on the data without normalization, since the data were designed to be well-scaled already.



(a) Close



(b) Far



(c) Wide

Figure 5-5. Synthetic Gaussian training (red and blue) and test (black) sets. 2D drawing of nD data, to scale. The solid circle indicates a radius of 1σ.

Table 5-1. Training data locations.

Name	Shape	Dimensions	Class	Location
Planes	hyperbox	3, 8	*Z*0	$z = 0$
			*Z*1	$z = 1$
Rings & Pluto	hypersphere	3, 8	*Inner*	$r = 1$
			Outer	$r = 2$
			Pluto	$r = 0.5, z = 12$
Normals	Gaussian	3, 8	A	$z = 0, \sigma = 1$
Close			B	$z = 1, \sigma = 1$
Far			B	$z = 2, \sigma = 1$
Wide			B	$z = 4, \sigma = 6$

5.3.1. Synthetic Data Experimental Observations

This section describes the observations about the trust metrics on the synthetic datasets described above. We expect that data from the *In*, *Mid*, and *Far* sets will have low trust by at least one metric, but not necessarily by all three. We value the fact that the metrics measure different

Table 5-2. Training data sizes.

Name	Dimensions	Classes #Points		
		Z0	*Z1*	
Planes	3, 8	1000	1000	
		Inner	*Outer*	*Pluto*
Rings & *Pluto*	3	400	1600	100
Rings & *Pluto*	8	20	1980	10
Rings & *Pluto* Dense	8	2.6k	83k	20
		A	B	
Normals *Close*	3, 8	1000	1000	
Normals *Far*	3, 8	1000	1000	
		Narrow	Wide	
Normals *Wide*	3, 8	500	1500	

Table 5-3. Test data location. *Far* outside the training data but closer to one set, far from the other. *In* is closer to that other set in *Rings*. For the other sets, *In* points are a subset of one class's distribution. *Mid* is between the two classes' distributions. A is from the distribution of the first class, and B is from the distribution of the second class. For planes, the *In* test data is already in-distribution, and the classes are symmetric, so no A and B distribution test sets are needed. For Normals, the *Far*, *In* and *Mid* test sets are $(D-1)$ hyperballs in the hyperplane with the given last coordinate. For Normals, the Indistribution test sets are Gaussians with the same offset and sigma as the training data. Each test data set has 100 points.

Name	Shape	Type	*Far*	*In*	*Mid*	Indistribution A	B
Planes	hyperbox	$z =$	-0.5	0	0.5		
Rings & *Pluto*	hypersphere	$r =$	2.5	0.5	1.5	1	2
Normals *Close*	hyperball	$z =$	-0.5	0	0.5	0	1
Normals *Far*		$z =$	-1	0	1	0	2
Normals *Wide*		$z =$	-2	0	2	0	4

things, and view this is a strength rather than a weakness. Our intention is that the synthetic data tests would provide greater understanding of how different the metrics are from each other, and what they are actually measuring. We discuss the experimental observations on each synthetic dataset in the following sections. In each section we describe the data in more detail, expectations and experimental observations.

Classifiers. We ran the experiments with two different classifiers. We used a Support Vector Machine (SVM) classifier, with a Radial Basis Function (RBF) kernel, which we assumed would be well suited for the rings and normals shapes. We also ran the experiments with a Random Forest (RF) classifier.

Clusterer. We used the HDBScan density based classifier. We used default parameters except a minimum cluster size of 10. When running it over the training data, we set the `prediction_data=True` keyword argument so as to speed up subsequent queries over the test data. We used the `approximate_predict()` function to evaluate the test points strengths without changing the clusters.

Metrics. Recall the *Training Proximity Metric* is measured by the cluster membership strength output by HDBScan. Higher strength is higher trust, assuming the cluster and the test point are of the same class. Recall the *Extrapolation Metric* is the signed distance to the convex hull boundary, calculated approximately through projections. A larger-magnitude negative distance is farther inside and more trustworthy. Recall the *Class Ambiguity Metric* is the distance to the decision boundary, approximated by the distance to the closest point we can find that is assigned a different class by the classifier.

5.3.1.1. *Planes*

See Figure 5-4a for a 2-dimensional representation of the datasets.

Data Description. The training points are in the $[0,1]^D$ hyperbox. Each class is in a $[0,1]^{D-1}$ sub-hyperbox. The $Z0$ class has the last coordinate set to 0, and the $Z1$ class has the last coordinate set to 1. The corners of the hyperbox for each class are explicitly added to the training data. This makes the convex hull deterministically equal to the entire (D-1) dimensional hyperbox, to avoid artifacts coming from the random point placement. The number of points in each class is slightly higher than 1000: 1004 in 3D, and 1128 in 8D. The test data have the same geometric shape and dimensionality as the training data. The *Far* test data are in the $[0,1]^{D-1}$ hyperbox with last coordinate -0.5; *In* has coordinate 0, and *Mid* has 0.5. See the *Planes* rows in Tables 5-1, 5-2, and 5-3 for specific values.

Expectations. We expect that *Mid* test points are very close to the decision boundary, about 0.1 or less, and would have extrapolation distances in [-0.5, 0]. We expect that the *In* test points are in the plane$Z0$ cluster and not outliers, and have a decision boundary score of about 0.5. We expect the *Far* test points are outliers and have an extrapolation distance of 0.5 or slighty more. We expect the 3D and 8D results to be comparable.

Observations. The numerical values for the trust metrics are shown in Table 5-4. Generally speaking, for the *Planes* with three features datasets the trust metrics matched expectations. For the *In* test set, some data points had a lower proximity strength indicating that even when data is drawn from the same distribution, there are cases where certain spaces are just not sampled very well. The ramification of this is demonstrated when a model is trained on data in a closed set and performs very well but performs poorly when deployed precisely because certain areas of the input space are not well represented. For the *Mid* test set, half of the test points belong to one cluster and the other half to the other and is represented with the lower proximity strength scores.

Table 5-4. Experimental observations on the *Planes* data

D	Metric	Test Scores		
		Far	*In*	*Mid*
3	Proximity	0.13	[0.64, $\sim$ 1]	[0.13, 0.13]
	Extrapol. Ave.	0.50	0	[-0.43, -0.0001]
	SVM decision	0.99	0.50	$\sim$ 0
	RF decision	0.99	0.52	$\sim$ 0
8	Proximity	[0.48, 0.63]	[0.65, 0.94]	[0.58, 0.64]
	Extrapol. Ave.	0.5	0	[-0.15, -0.001]
	SVM decision	1.01	0.54	$\sim$ 0
	RF decision	[1.01,1.04]	[0.51, 0.55]	$\sim$ 0

It was unexpected that for the higher dimensional data, 8D vs. 3D, the training proximity strengths are much higher for the out-of-distribution *Mid* and *Far* test data, and lower for the in-distribution *In* data. Being lower for the in-distribution data is less troubling than being higher for the out-of-distribution data. Overall, we have little insight into how to interpret the strength values, how to compare the strengths between scenarios, or what a reasonable threshold might be.

For the case with eight features, there is a change in proximity strengths that demonstrate more uncertainty. For the *Far* set of points, the values are around 0.5, all belonging to the cluster of the plane closest to the far points. For the *Mid* set of points, the proximity values increase. While the values are expected, the change in values represents a challenging problem in increasing the feature size and requiring more data points to fully sample the space at equal density.

For both 3D and 8D, it was unexpected that the RF classifier would have such a crisp decision boundary, and for it to be midway between the two classes, matching the location for the SVM.

The convex hull distances being closer to 0 for 8D compared to 3D makes sense, because there are more coordinates so it is more likely at least one coordinate is close to 0 or 1, the convex hull boundary.

The decision boundary distances are slightly higher in 8D than 3D. It may be because in 8D the data are spread out so the first different-class point is farther away, and it is harder to search to find the closest decision boundary point.

5.3.1.2. *Rings* and *Pluto*

See Figure 5-4b for a 2-dimensional representation of the datasets.

Data Description. The training points are on a $(D-1)$ sphere, the surface of a D ball. For the *Rings* datasets, the test data have the same geometric shape and dimensionality as the training data; e.g. the rings' test data are also from hyperspheres, with the same center, just different radii. The *Inner* class has radius 1, and the *Outer* class radius 2; both are centered at the origin. The *Pluto* class has radius 0.5 and center offset from the origin by 12 units in the last coordinate. For the *Rings* and *Pluto* training sets, the same data points are used for the two rings; the only difference is the inclusion of the far sphere of points making up the *Pluto* class. This avoids any artifacts that might arise from changes to the random placement of the ring points. For 3D, *Inner* has 400 points, *Outer* 1600 points, and *Pluto* 100 points. For 8D, the number of ring training points remains at 2000, but *Inner* has 20 points, *Outer* 1980 points, and *Pluto* 10 points; the relative density of each of the spheres is about the same. (This was done since we are using a density-based clustering algorithm and varying densities does cause some problems as the number of dimensions increases—a curse of dimensionality). For the 8D dense set, *Inner* has 2.6k points, *Outer* has 73k points, and *Pluto* has 20 points. Thus, the number of points relative to the surface areas of the spheres is the same for the 8D Dense and 3D sets. The *Far* test data have radius 2.5, *Mid* radius 1.5, and *In* radius 0.5; all are centered at the origin. See the *Rings* and *Pluto* rows in Tables 5-1, 5-2, and 5-3 for specific values.

Expectations. *In*, *Mid* and *Far* extrapolation distances should be about -1.5, -0.5, and 0.5. None of the data points should belong to a cluster. *Mid* should have very small decision boundary distance, close to 0, while *In* and *Far* should have distance about 1. We expect the inclusion of the *Pluto* class to have little effect on the metrics, except some fraction of the *Far* points should become inside the convex hull, with distance up to about -1. The 8D versions should have the same behavior, up to the data being more sparse so there are more opportunities for randomness to manifest.

Observations. Results are shown in Table 5-5. The variance across the 100 test points for most of the metrics was not large, so we usually only report the average values. The exception is we report the min and max separately for the Extrapolation distances, because these varied significantly within a test set. Unfortunately, including *Pluto* in the training data causes the HDBScan clusterer to put the inner and outer training points into the same cluster. Then all the *Far*-*Mid*-*In* test points belong to that cluster with strength 1. Our output being sensitivity to the inclusion of spurious data like this is undesirable, yet insightful, shedding light on how effects of the outlier data is not localized, but has global ramifications.

Rings 3D *Mid* Proximity was identical to *Pluto* 3D *Mid* Proximity: 65 points were in cluster 1 (outer) with average strength 0.66, and 35 points were in cluster 0 (inner) with average strength 0.61.

Table 5-5. Experimental observations on the *Rings* and *Pluto* datasets. The average Proximity values include outliers as zeros.

D	Train	Metric	Test Scores				
			Far	*In*	*Mid*	*Inner*	*Outer*
3	*Rings*	Proximity	0.58	0.62	0.64	0.96	0.98
		Extrapol. Ave.	0.51	-1.49	-0.49	-0.99	0.01
		[min,max]	[0.50,0.52]	[-1.50, -1.48]	[-0.50, -0.49]	[-1.00, -0.98]	[0.00, 0.02]
		SVM decision	1.5	0.50	0.51	0.09	1.00
		RF decision	1.18	0.58	0.28	0.14	0.72
3	& *Pluto*	Proximity	0.58	0.62	0.64	0.97	0.98
		Extrapol. Ave.	0.33	-1.52	-0.57	-1.09	-0.15
		[min,max]	[-0.68,0.52]	[-1.87,-1.49]	[-1.59, -0.49]	[-1.75, -0.99]	[-1.51, 0.02]
		SVM decision	1.51	0.50	0.51	0.09	1.0
		RF decision	1.18	0.58	0.27	0.15	0.70
8	*Rings*	Proximity	0	0	0.23	0	0.11
		Extrapol. Ave.	0.28	-1.29	-0.50	-0.90	-0.10
		[min,max]	[-0.16, 0.69]	[-1.38,-1.21]	[-0.74, -0.27]	[-1.10, -0.75]	[-0.40, 0.29]
		SVM decision	1.96	0.79	1.17	0.85	1.55
		RF decision	1.96	0.78	1.16		
8	& *Pluto*	Proximity	0	0	0	0	0.02
		Extrapol. Ave.	0.26	-1.30	-0.52	-0.92	-0.13
		[min,max]	[-0.47, 0.69]	[-1.41, -1.21]	[-0.79, -0.27]	[-1.25, -0.75]	[-0.88, 0.29]
		SVM decision	1.96	0.79	1.17	0.85	1.54
		RF decision	1.96	0.78	1.16		

For *Rings* 3D, and *Pluto* 3D, with SVM classifier, all *Mid* points were classified as *Outer*. The data are consistent with a roughtly spherical decision boundary at radius 1.05, just beyond the training set *Inner* class.

For *Rings* 3D and RF, of the *Mid* points, 95 were *Outer*, and 5 *Inner*. Of the *Inner* test points, 4 were classified as *Outer*. For *Pluto*, again 95 *Mid* and 4 *Inner* test points were *Outer*. The decision boundary distances were slightly different when including *Pluto*, but not significantly. The data are consistent with an undulating spherical decision boundary, at radius 1 to 1.5.

For *Rings* 8D Proximity, all *Far* and *In* points are outliers. *Mid* had 77 outliers, 22 in cluster 1 with strength 0.99, and 1 in cluster 0 with strength 1. Test points from the *Inner* distribution were all outliers. For the *Outer* distribution test points, 89 were outliers, 10 were in cluster 1 with strength 0.99, and 1 was in cluster 0 with strength 1.

For *Pluto* 8D, all *Far*, *In*, *Mid*, and *Inner* points are cluster outliers. 98 *Outer* points are outliers, 1 was in cluster “3” with strength 0.99 and 1 was in cluster “5” with strength 0.97.

For *Rings* 8D, the decision boundary metric had high variance for *Mid*, with the SVM having range [0.70, 1.51] and average 1.17; and RF having range [0.67, 1.52] and average 1.16. For both

Rings 8D and *Pluto* 8D, all *Far*, *In*, *Mid*, *Inner*, and *Outer* test points were classified by the SVM and the RF as *Outer*. Including *Pluto* did not significantly change the classifications or decision boundary distances.

For a fixed classifier and dimension, the data are consistent with the decision boundary resembling some sphere with noise. However, changing the classifier (SVM vs. RF), or using the same classifier and changing the dimension (3D vs. 8D), often dramatically changes the decision boundary's apparent radius. In some cases it is very near the inner ring, sometimes near the outer ring, and sometimes half-way between. The computed distances have high variance. In 8D the descision boundary distances are a bit larger than 3D, as for planes as discussed in Section 5.3.1.1.

For training proximity strength, in 3D the results are at least self-consistent: high strengths near 1 for the in-distribution points, and strengths about 0.6 for the test rings radius 0.5 away from one of the training classes. In 8D, almost all the training data and test data are outliers. The exception is 4 of the test points in the outer-ring distribution have high strengths with the outer-ring class cluster, and 1 of them has high strength with the inner cluster.

All the extrapolation scores make sense and are as expected. In 8D the variance is high, but the average is about as expected for all test cases.

Including the *Pluto* training points had little effect, as we expected (and hoped). The only significant difference is that in 8D, *Mid* test data, 22 / 100 points are considered part of a cluster if *Pluto* is not present, whereas all of the *Mid* points are outliers if pluto is part of the training data. It appears the HDBScan cluster is sensitive to including a cluster of outlying points.

In 8D rings, all test points were classified as *Outer*, perhaps because the *Inner* training data were too sparse. This motivated the generation of the 8D dense *Rings* & *Pluto*.

5.3.2. *Examination of Dense* **Rings & Pluto** *8D*

Here we investigate a denser version of the *Rings* and *Pluto* datasets. One of the major lessons learned is that the dimensionality of the data dictates the amount of points that are needed to understand the training space. If there are too few points, determining a good representation of the space is difficult. Thus, in some of our studies we increase the number of data points to created denser coverage of the space. Of course, this is not possible when working with real-world datasets. That there might be too few points to describe the training space is an important caveat to keep in mind when dealing with real-world datasets, and it *may* contribute to the susceptibility to adversarial attacks of many high-dimensional models.

The *Rings* and *Pluto* 8D *Dense* training set has about the same number of points per unit hyper-area as the 3D *Rings* and *Pluto* training sets, so one might hope to get similar neighborhood trust scores. In practice it fell short of achieving this hope, but it did improved the results compared to the 8D non-dense sets.

For *Rings* Proximity, *Far* had 3 outliers, and 97 points belonged to cluster 1; *In* had 100 in cluster 0; *Mid* had 23 in cluster 0 (inner) with average strength 0.73, and 77 in cluster 1 (outer) with

D	Train	Metric	Test Scores				
			Far	*In*	*Mid*	*Inner*	*Outer*
8	*Rings*	Proximity	0.72	0.87	0.85	0.94	0.95
		Extrapol. Ave.	0.13	-1.49	-0.68	-1.08	-0.26
		[min,max]	[-0.17, 0.48]	[-1.56, -1.41]	[-0.90, -0.46]	[-1.23,-0.89]	[-0.57, 0.03]
		SVM decision	0.97	1.1	0.09	0.62	0.45
		RF decision	1.5	0.47	0.39	0.08	0.94
8	& *Pluto*	Proximity	0.99	1	1	1	1
		Extrapol. Ave.	0.11	-1.5	-0.7	-1.1	-0.29
		[min,max]	[-0.63, 0.48]	[-1.6, -1.41]	[-0.97,-0.46]	[-1.52,-0.89]	[-1.08,0.03]
		SVM decision	0.97	1.1	0.1	0.62	0.45
		RF decision	1.5	0.47	0.39	0.08	0.94

Table 5-6. *Rings* & *Pluto* 8D dense experimental data.

average strength 0.89. For *Pluto* Proximity, *Far*, *In* and *Mid* test points were all assigned to cluster 1 with very high strength.

For the SVM decision boundary, all *Mid* points were class *Inner*, regardless of *Pluto*. The other test points were assigned the expected class by the SVM.

The RF assigned the same class to test points regardless of whether *Pluto* was included in the training data, and the decision boundary distances were the same. All *Far*, *Mid*, and *Outer* test points were classified as *Outer*. All *In* points were *Inner*. However, of the *Inner* test points, 26 were classified *Outer*, and 74 *Inner*; note the very small average distance to the decision boundary, 0.08. The data are consistent with the RF decision boundary being roughly spherical with radius undulating between 1 and 1.1.

5.3.2.1. Normals

See Figure 5-5 for a 2-dimensional representation of the datasets.

Data Description. The Normals training family consists of *Close*, *Far*, and *Wide*. There are two classes. Class A is a normal (Gaussian) distribution with $\sigma = 1$ centered at the origin. The test data lie in hyberballs of one less dimension, in a hyperplane. For the *Close* and *Far* sets, class B is the same normal distribution, $\sigma = 1$, offset 1 and 2 units in the z (last coordinate) direction. For the *Wide* set, class B has $\sigma = 6$ and is offset 4 units. The normals overlap significantly in the *Close* and *Far* sets, and in the *Wide* set class A appears inside class B. In *Close* and *Far* there are 1000 points in class A and B; in *Wide* there are 500 and 1500 points.

Expectations. For two Normals *Close* and *Far*, the test points *Far*, *In*, and *Mid* should all be interpolations with distances in [-1 , 0], depending on the random sampling. We do not know if the clusterer will generate one or two clusters, since the normals overlap fairly strongly, especially for *Close*. *Mid* points are close to being part of both clusters, so if there are two clusters, the strengths should be the same. *In* should be part of the cluster of class A, and especially *Far* should be part of cluster A (or outliers).

We expect the decision boundary trust to be small, since the distributions overlap, and the classes are not well separated. Perhaps for class ambiguity trust *Far* > *In* > *Mid*. For two normals *Wide*, *Far*, *In*, and *Mid* should be interpolations, up to random quirks. *In* should be in the cluster of class A (*Narrow*). *Mid* might be in the cluster of class B (*Wide*) or outliers. *Far* should be outliers. *Mid* and *Far* may be close to the decision boundary. *In* should be farther from the decision boundary than those other two. The 8D versions of should have roughly the same behavior as 3D, but, as with *Rings*, the data are sparser so we do not expect the clusterer to work well. The decision boundary trust might go up with the larger distances between points.

Train	Metric	Test Scores				
		Far	*In*	*Mid*	A	B
Close	Proximity	0.98	0.99	1.00	0.99	0.97
	outliers	75	38	23	86	82
	Extrapol. Ave.	-1.9	-2.3	-2.4	-1.5	-1.5
	[min,max]	[-2.1, -1.58]	[-2.6, -1.9]	[-2.9,-1.9]	[-2.7, 0.21]	[-2.9, 1.0]
	SVM decision	0.20	0.19	0.04	0.29	0.34
	[min,max]	[0.04, 0.40]	[0.05, 0.35]	[1e-3, 0.10]	[0.01, 1.2]	[0.01, 2.6]
	A, B	100, 0	100, 0	82, 18	71, 29	29, 71
Far	Proximity	0.93	0.93	0.94	0.91	0.94
	outliers	7	20	3	69	68
	Extrapol. Ave.	-2.4	-2.2	-2.5	-1.5	-1.5
	[min,max]	[-2.9,-1.9]	[-2.6, -1.8]	[-3.0,-1.9]	[-2.9, 0.21]	[-2.8, 1.0]
	SVM decision	0.78	0.32	0.05	0.52	0.5
	[min,max]	[0.38, 1.12]	[0.16, 0.51]	[0.03, 0.27]	[8e-3, 1.9]	[0.03, 3.1]
	A, B	100, 0	100, 0	100, 0	84, 16	10, 90
Wide	Proximity	0.45	0.48	0.86	0.6	0.36
	outliers	0	0	0	0	97
	Extrapol. Ave.	-9.1	-13.0	-11.0	-11.1	-8.5
	[min,max]	[-9.2, -8.9]	[-13.8, -12.8]	[-11.2, -10.8]	[-13.1, -8.9]	[-15.5, 6.2]
	SVM decision	0.9	1.1	0.7	0.8	7.4
	[min,max]	[0.46, 1.2]	[0.69, 1.3]	[0.4, 1.2]	[0.14, 1.4]	[0.22, 25.5]
	A, B	100, 0	100, 0	100, 0	100, 0	2, 98

Table 5-7. 3D Normals experimental data. Here we report the Proximity strength averages *excluding* the outliers. For the SVM and RF classifiers, we report the number of test points assigned to each of class A and B.

Observations. Results on the normal datasets are provided in Tables 5-7 and 5-8. For training set, *Far* test points, the RF did not separate the points well and had about half in each class, with a very small distance to the decision boundary. This is similar to how RF behaved for the *Mid* test points as well.

Train	Metric	Test Scores				
		Far	*In*	*Mid*	A	B
Close	Proximity	1.0	1.0	1.0	1	1
	outliers	92	52	29	99	99
	Extrapol. Ave.	-2.1	-2.2	-2.1	-0.8	-0.8
	[min,max]	[-2.2, -1.7]	[-2.5, -1.8]	[-2.5, -1.6]	[-1.9, 1.5]	[-2.1, 1.2]
	SVM decision	1.1	0.7	0.13	1.1	1.0
	[min,max]	[0.46, 1.4]	[0.40, 1.3]	[2e-3, 1.1]	[5e-4, 2.6]	[0.01, 2.1]
	A, B	100, 0	100, 0	21, 79	71, 29	27, 73
Far	Proximity	1.0	1.0	1.0	1	0
	outliers	35	51	37	99	100
	Extrapol. Ave.	-2.2	-2.2	-2.2	-0.8	-0.8
	[min,max]	[-2.5, -1.8]	[-2.5, -1.9]	[-2.5, -1.9]	[-1.9, 1.5]	[-2.1, 1.2]
	SVM decision	1.5	1.2	0.1	1.4	1.3
	[min,max]	[1.0, 2.0]	[0.8, 1.7]	[0.01, 1.1]	[0.02, 3.4]	[0.09, 2.8]
	A, B	100, 0	100, 0	0, 100	87, 13	14, 86
Wide	Proximity	0	0	0	0	0
	outliers	100	100	100	100	100
	Extrapol. Ave.	-10.2	-14.0	-12.1	-11.8	-4.0
	[min,max]	[-10.5, -10.0]	[-14.2, -13.8]	[-12.3, -11.9]	[-13.6, 10.0]	[-11.4, 7.2]
	SVM decision	5.2	4.1	5.5	4.1	11
	[min,max]	[4.4, 6.0]	[3.6, 4.7]	[4.9, 6.0]	[2.2, 6.0]	[1.3, 23]
	A, B	100, 0	100, 0	100, 0	100, 0	0, 100

Table 5-8. 8D Normals experimental data. Here we report the Proximity strength averages *excluding* the outliers.

For 8D *Close* Normals, Proximity metric, *Far* test points, the 8 non-outliers were split between 2 in cluster “0” and 6 in cluster “1.” For the *In* test points, the 48 non-outliers were split between 25 in cluster “0” and 23 in cluster “1.” For the *Mid* test points, the 71 non-outliers were split between 22 in cluster “0” and 49 in cluster “1.”

For 8D *Far* Normals, Proximity metric, *Far* test points, the 65 non-outliers were split between 36 in cluster “0” and 29 in cluster “1.” For the *Mid* test points, the 63 non-outliers were split between 37 in cluster “0” and 26 in cluster “1.”

For 8D *Wide* Normals, HDBScan formed no clusters, so all test points were automatically outliers. For RF, all *Far*, *In* and *Mid* points were classified as class A (*Narrow*). Unfortunately, the decision boundary distance was *larger* for *Far* and *Mid*, indicating more trust than for the *In* points, but the *In* points are located within the class A (*Narrow*) distribution, and *Far* and *Mid* are not. A possible explanation is that since the class B (*Wide*) normal 1-sigma ball contains the class A (*Narrow*) normal 1-sigma ball, there may be many class B (*Wide*) test points near the *Mid* test points. Thus either the classifier or the metric failed to perform as desired in this case.

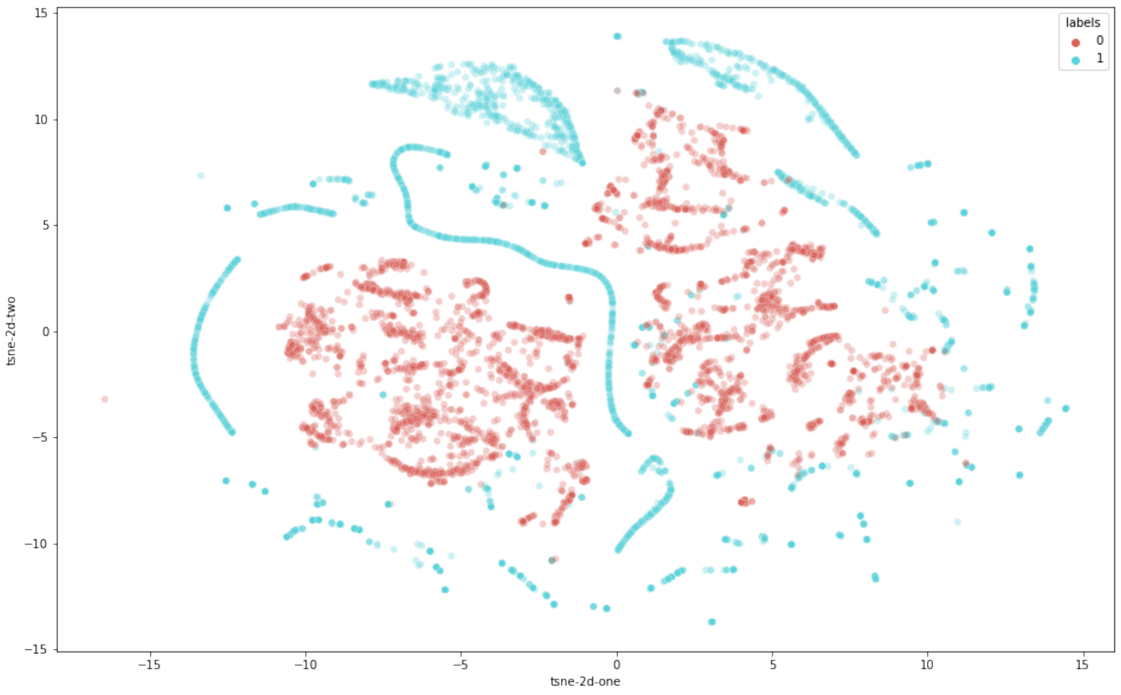
5.4. Experimental Results on PDFrate

We examine the GeoTrust metrics using the task of identifying malicious PDFs where the features are not intuitive and false negative can be very costly to an organization. We use the PDFrate dataset [108] following their implementation of using a random forest (RF). There are 2 datasets, "Contagio" which is a large academic dataset that has been used in several studies and "University" which was collected from monitoring HTTP and SMT traffic and can be considered operational data. We also examine EvadeML which is a set of detection evasion data points: all are malicious PDFs that are evading detection and represent a high-consequence risk if they go undetected [26]. The RF classifier performs well on the training set (here, we use all of the data for training) and about half of the EvadeML (53%) ans well as the University data (57%) bypass detection, representing a serious problem for a cyber defender.

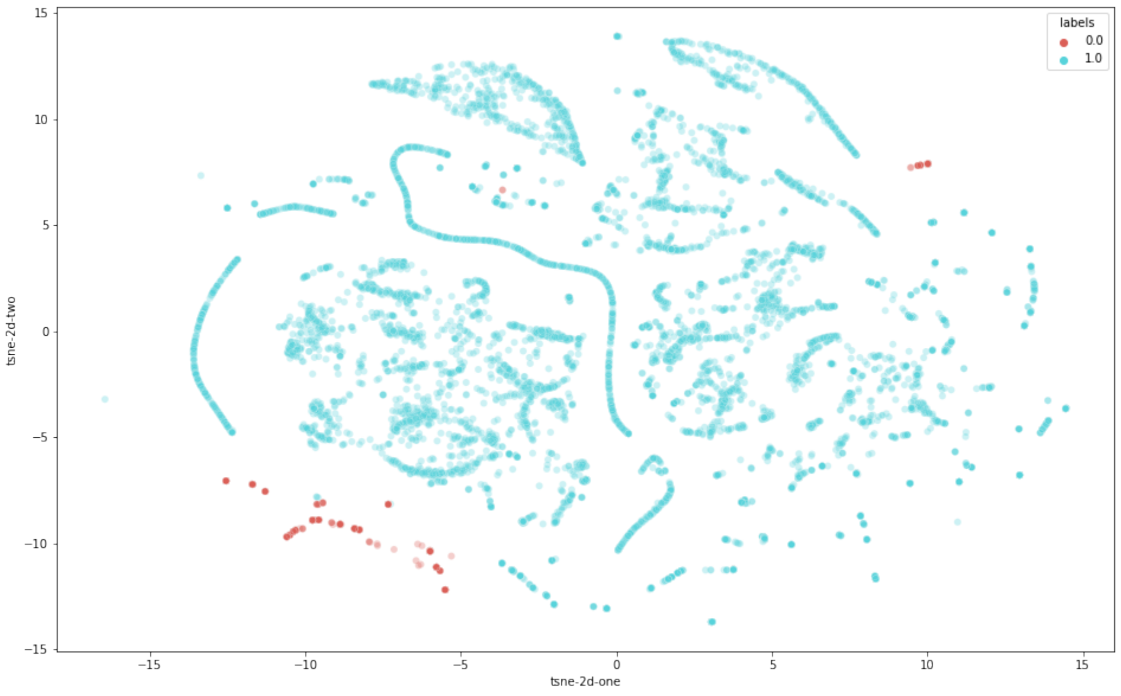
Results for the GeoTrust metrics using the RF classifier are summarized in Table 5-9 and yield interesting insights. (1) The Training Proximity Metric identifies all of the evasion attempts as outliers. This can be seen visually in Figure 5-6 where the data is projected to 2 dimensions using t-SNE. Visualizing the data makes it apparent that there is good class separation between the malicious and benign data points (left) and that the evasion attacks are not near the training data (right). (2) The Extrapolation Metric identifies all of the evasion data points as extrapolations. This was surprising as we suspected that adversarial attacks would be designed as interpolation in an attempt to hide but this examination indicates that the attacks are actually occurring in dimensions and at magnitudes which make them extrapolations. This is also apparent in the visualization. (3) All of the data points have similar distances to the decision boundary. Again, based on our visualization, this is intuitive. However, we would have suspected that the adversarial attacks would have appeared closer to the classification boundary but rather form their own clusters. (4) The RF classifier does not generalize well to the operational data. This is a common problem when deploying learned models that have been developed on hand curated data sets. They often perform poorly when deployed to operational data. The GeoTrust metrics indicate that they too are significantly different from the training data, and the classifier performs poorly—achieving slightly better accuracy than EvadeML.

5.5. Comparison with Out-of-Distribution Methods

We compared our out-of-distribution detection methods against several leading publications. Specifically, we test against baseline (maximum softmax probability) [43], outlier exposure [44], and DeepMCDD [62].



a



b

Figure 5-6. t-SNE projection of the PDFrate data. a) Malicious vs benign. The 0 label represents the benign class while 1 represents malicious. It can be seen that there is some class separation. b) PDFrate vs Evade. The 0 label represents the evasion attack points and the 1 label represents the original PDFrate data. Visually, it is easy to see that the evasion attacks occur in less populated regions of the input space.

Table 5-9. Summary of the trust metrics on the PDFrate data set. The average Training Proximity Metric is the strength values returned by HDBscan where 1 indicates a strong relationship with a cluster and 0 represents an outlier. The Extrapolation Metric indicates the distance from a convex hull around the training data. A value of 0 indicates that the test point is on the convex hull. The Class Ambiguity Metric indicates the distances from the classification boundary.

Trust Metric	Contagio	University	EvadeML
Avg Training Proximity Metric	0.742	0.091	0
Max Extrapolation Metric	0	1.118	0.729
Min Extrapolation Metric	0	0	0.055
Avg Extrapolation Metric	0	0.002	0.084
Max Class Ambiguity Metric	0.342	0.330	0.273
Min Class Ambiguity Metric	0	0	0
Avg Class Ambiguity Metric	0.019	0.002	0.017

5.5.1. Maximum Softmax Probability

In [43] they present a simple baseline for detecting if an example is misclassified or out-of-distribution that utilizes probabilities from softmax distributions. The basis is on the fact that correctly classified examples tend to have greater maximum softmax probabilities than erroneously classified and out-of-distribution examples, allowing for their detection. To get the baseline score, they retrieve the maximum/predicted class probability from a softmax distribution and thereby detect whether an example is erroneously classified or out-of-distribution. For “In,” they treat the in-distribution, correctly classified test set examples as positive and use the softmax probability for the predicted class as a score, while for “Out” they treat the out-of-distribution examples as positive and use the negative of the aforementioned probability.

5.5.2. Outlier Exposure

The outlier exposure method uses outside datasets to make classifiers robust against bad inputs. In [44], they train models to detect unmodeled data (ood) by learning cues for whether an input is unmodeled. Specifically, they found that the model can learn effective heuristics for detecting out-of-distribution inputs by exposing the model to OOD examples that are entirely disjoint from test-time data. Additionally, they train their model without tuning parameters to fit specific types of anomaly test distributions, so their results are not directly comparable with other OE-based method results.

5.5.3. DeepMCDD

DeepMCDD [62] is an effective method for OOD detection. This method aims to find a spherical decision boundary for each class instead of focusing on the linear decision boundary that partitioning its latent space into multiple regions. The key idea is try to make the latent representations of data sample in the same class to form an independent sphere of minimum column. That means they can model discriminative class-conditional distributions by learning multiple Gaussian distribution instead of the hyperspheres. Based on learned distribution, they calculate the class-conditional probabilities of input X from each class. Distance function D_k based on k-th class conditional distribution is:

$$\begin{aligned} D_K(\mathbf{x}) &= -\log P(\mathbf{x}|y=k) \\ &= -\log \mathcal{N}(f(\mathbf{x};\mathcal{W}) \mid \mu_k, \sigma_k^2 I) \\ &\approx \frac{||f(\mathbf{x};\mathcal{W}) - \mu_k||^2}{2\sigma_k^2} + \log \sigma_k^d \end{aligned}$$

The class label of input sample is predicted based on the highest posterior probability.

5.6. Implementation

The Synapse deep learning cluster was used for the testing platform of the different methods. After downloading the files from their respective Github repositories (DeepMCDD[1], MSP/OE[2]), they were tested for use-ability. Specifically, we ran the given examples in the repositories and checked that the results matched what was reported in the papers.

After verifying the methods were working as reported, we tested the methods on detecting malicious PDFs from the PDFrate dataset [108]. The PDFrate dataset[3]only had one test, with the Contagio dataset being used as in-distribution and the EvadeML [126] dataset as out-of-distribution.

The outlier exposure github repository provided network architecture for the image tests. In order to create a neural network that was able to accurately classify a PDF as malicious or not, we completed neural architecture search on the Contagio dataset using Auto-Keras[4]. After getting the best performing hyperparameters and model architecture, the network was built out in PyTorch and trained on the Contagio dataset. After training, the network was able to classify the Contagio dataset as being malicious or not with 98% accuracy.

5.7. Results

Results are shown in Table 5-10 for the compared OOD methods on PDFrate. *We first note that all of these results are preliminary*. It is difficult to do a direct comparison since the GeoTrust metrics are designed provide a measure of trust rather than a binary decision. However, for

[1]https://github.com/donalee/DeepMCDD
[2]https://github.com/hendrycks/outlier-exposure
[3]https://github.com/csmutz/pdfrate/
[4]http://autokeras.com/

completeness, we include these results and allow extrapolation. We observe that all three methods are able to achieve some success in detecting EvadeML.

Table 5-10. Contagio Dataset Results. TP: correctly predicted ood at recall=0.95

ID	Method	OOD	FN	TN	FP	TP	AUROC	AUPR
Classifier Contagio	MSP	Evademl	**840**	1271	8729	**15937**	79.05	86.70
	OE	Evademl	**840**	**2815**	**7185**	**15937**	58.99	**91.01**
	DeepMCDD	Evademl	858	28	9972	15919	**96.28**	58.29

If we take the liberty to extrapolate, even just using the Training Proximity Metric, we can conclude in this case that GeoTrust is able to identify *all* of the adversarial attacks. Likewise, just using the Extrapolation Metric, all of the EvadeML data points are outside of the convex hull. Of course, this merits additional investigation, but initial results are promising and coincide with many bold claims in the literature the nearest-neighbor methods are robust to adversarial attacks [84, 121, 53].

5.8. Future Work

HDBScan clustering. The 8D rings vs. pluto, cluster proximity failed to be informative about classification trust in both the original (sparse) and the dense variation, finding no clusters in the first, and one giant cluster containing both inner and outer classes in the other. The fact that throwing in 20 outliers (pluto) into a pool of 85k points changed the clustering reduces our confidence that HDBScan's strengths can be meaningfully interpreted across scenarios. (HDBScan's clustering strengths can often distinguish between nearby and far-away test points for a fixed set of training data. What is unclear is whether the strengths for one set of training data can be compared to the strengths for another set of training data.)

Preliminary research indicates that tuning the HDBScan parameters may be helpful. Specifically, using a minimum cluster size of 10 is clearly not the best for all data. As base-level heuristic, adjusting the minimum cluster size to get a "reasonable" amount of outliers seems to be be a first pass. Another possibility to examine the trust metrics is a lower-dimensional space. We find this line of research promising as there exists a niche research line examining the robustness of distance-based measures [84, 121, 53]. One major hurdle is the curse of dimensionality requiring significantly more points as the dimensionality increases.

Class-specific Clustering. We conjecture that additional information can be gained by clustering the classes of a training set separately, and computing the neighborhood metric on each class-clusterer. For example, if a test point is classified as A but its strength of belonging to every class A cluster is low, but it strongly belongs to a class B cluster, then we would have lower trust in its classification as A.

However, initial experiments were not successful. We tried the *Rings* 3D data, building one HDBScan clustering based on just the *Inner* points, and one on just the *Outer* points; we call these the *trained* clusterers. Each trained clusterer calculated all *Far*, *In*, and *Mid* point strengths as outliers, not belonging to any cluster. While not informative, this is also not unreasonable. However, even the *in distribution* test points were calculated as outliers, which we consider a failure.

It appears that having two distinct distributions helps the clusterer identify them. For future work, we could try parameter tuning. We could assume that the class-specific clusterer should cluster the majority of the training points, and tune the parameters until this occurs.

For future work, we also wish to consider clusters trained pair-wise. E.g., train one clusterer on *Inner* and *Pluto*, one on *Outer* and *Pluto*, and one on *Inner* and *Outer*. For each cluster, identify it as the class of the majority of its points. (If a cluster has no clear majority, this already informs us we should have less trust in the data, since the classes are mixed.) Then, e.g., for a test point classified as *Inner*, compute its strengths on the *Inner-Pluto* clusterer and the *Inner-Outer* cluster. We would have higher trust if the strengths are stronger for the *Inner* class clusters. It seems like what we really care about it density in a space. In ML, we often are so concerned about class imbalance, but density may be more important as shown in the 2 rings experiment with 2.6k and 83k points in the different classes.

Training Class Overlap. We had not specifically considered training class overlap in our initial development of the trust metrics. We hypothesize that we can easily convey this measuring class-specific convex hulls and determining if a data point is simultaneously in multiple class specific convex hulls. A similar notion is applicable for the class-specific clustering. Of course, for this to be effective, we would need to preprocess the data to remove outliers (possibly mislabelled data points).

5.9. Conclusion

The GeoTrust metrics for measure the trust of machine learning classifications and encompass (1) nearness to training points, (2) interpolation/extrapolation of training points, and (3) distance to decision boundary. While we demonstrated the utility of such an approach, there are limitations. As the number of dimensions increases, distances are less meaningful and more expensive to compute; these are part of the "curse of dimensionality." As part of future work, we are also looking to extend these methods to more traditional image processing and text spaces, where distances between images, for example, is difficult to calculate.

6. USER STUDY RESULTS

This chapter details the user studies that we conducted:

1. **User Study 1**: General user study to determine initial observations on the impact of explanations and to determine the best ways to present explanations in terms of the number of features and how to represent them.
2. **User Study 2**: Specific application of explanations integrated into an real-world workflow (in this case an Enterprise Cybersecurity operation).
3. **User Study 3**: Follow up research comparing different types of users (model developers and maintainers versus cyber analysts).

In user studies 2 and 3, the number of participants is low and thus qualitative analyses are used. Our overall conclusions are relatively negative for MLE. In many cases, the explanations are not used and do not appear to increase a user's trust in an ML model. We propose that alternative methods should be researched as simply providing important features as proposed in much of the literature is insufficient.

6.1. Sage Advice? The Impacts of Explanations for Machine Learning Models on Human Decision-Making in Spam Detection [1]

The use of machine learning (ML) algorithms to aid human decision-making is growing in popularity and is increasingly used in high-consequence applications. As such, it is important to have confidence that the ML model appropriately models the given task. ML explainability [16] is an increasingly popular topic in ML that seeks to build confidence in a ML model by providing explanations for individual predictions in the form of additional information that indicates why the prediction was made. An end user then uses the provided information for improved decision-making. However, how to present the explanation and the efficacy of an explanation on human decision-making is lacking.

There is much debate in the explainable ML community regarding what constitutes a good explanation (e.g., [6, 74]). Two possible definitions include: "the degree to which a human can understand the cause of a decision" [74] or the degree to which a human can consistently predict the model's result [54]. Additionally, Molnar [76] suggests in his book that some properties of good explanations are that they should be accurate, consistent, stable, complete, and

[1]Mallory C. Stites, Megan Nyre-Yu, Blake Moss, Charles Smutz, and Michael R. Smith. Sage Advice? The Impacts of Explanations for Machine Learning Models on Human Decision-Making in Spam Detection. In *International Conference on Human-Computer Interaction*, pp. 269-284. Springer, Cham, 2021.

“comprehensible.” Molnar [76] has posited that “human-friendly” explanations are: contrastive (i.e., why was this output chosen instead of another), selected (i.e., not the full list of causes for a decision), social (i.e., are designed with the target audience in mind), and focused on the abnormal (i.e., counterfactuals, or what would need to change about the input the change the output). However, being able to assess whether the end user of a system correctly understands the cause of a decision depends on the creator of the model also being able to understand the cause of those decisions, which is not often the case, especially with complex models.

A growing body of work has begun to investigate the attributes that make an explanation useful from the end-user’s perspective. Because most ML applications are created to help a human operator complete a task or make a decision, these studies investigate the extent to which explanations from the ML improve the human’s ability to accurately and efficiently complete their intended task. Although the individual tasks used differ widely, most of these studies start by measuring how well users can perform the task without model output (as a baseline). They then manipulate some aspect of model output shown to users (i.e., the presence of a suggested classification or correct answer; model confidence; values on features that were important in the model decision; feature importance) in order to measure whether the additional model information improves performance relative to baseline. Most previous studies have found that providing a decision from the model improves performance relative to having no model decision [37, 56, 60, 22]. Presenting information about model confidence and/or overall model accuracy also improves performance [60]. Providing users with more information from the model leads to greater reliance on the model’s decisions, even when the information provided is random or incorrect [17, 60]. There have been inconsistent findings as to whether the addition of Shapley Additive Explanation (SHAP) values, showing the importance of each feature to the model’s decision, affects participant performance. At least one study found that providing SHAP values in addition to feature values and model confidence did not affect accuracy in a classification task [123], whereas a different study shows that giving users feature importance values did improve accuracy in a model judgement task [114]. Showing users graded highlighting of important terms (in a text-reading task) or pixels (in a visual search task) based on model confidence of importance of those regions improved decision-making relative to single-color highlighting that ignored model confidence [56, 60]. Showing users more complex explanations may decrease their satisfaction with the model [79], and cause a drop in task performance [51]. Although user satisfaction is not a direct impact on performance accuracy, it may play a role in model adoption or trust.

However, there are a few gaps relating to these studies that make it difficult to confidently generalize findings or draw consistent conclusions. For example, many of the previous studies do not provide information about the experiment-wise accuracy of the model used. Many previous studies provide users with real output from models, which tend to be highly accurate (e.g., 87% accurate; [60]), meaning that most of the model predictions being shown to users in these studies are correct. This could skew results by seeming to show that presenting participants with model in-formation always improves performance, when what the results may really show is that providing predictions from a reliable model improves performance. Most studies also fail to describe the nature of the errors the model made. Based on findings from the trust in automation literature, it has been shown that if an automated system makes errors that the human analyst can easily recognize, this will degrade their trust in the system (for reviews, see Hoff and Bashir [46];

Schaefer, Chen, Szalma, and Hancock, [101]). If certain types of decision errors carry different consequences in the operational task being performed—like missing a weapon in a bag screening at a TSA checkpoint, or a malicious email making it through a firewall—users might be more reluctant to trust certain decisions from a model due to the high cost of an incorrect decision. Indeed, Perkins, Miller, Hashemi, and Burns [88] have found that people rely less on an automated system for decision-making when greater risk is involved. For many applications, the ML explanation becomes increasingly important in cases for which the model has low confidence. Human operator involvement is heavily biased toward the difficult (low accuracy) cases, which are likely ambiguous in some way. For this reason, it is also important to account to task differences, as well as individual familiarity or expertise with the task. For example, Feng and Boyd-Graber [30] found that experts and novices in the game QuizBowl benefitted from different types of assistance given by a ML model. In general, people are less likely to use automation to complete a task if they are highly skilled at the task already [100] or have high self-confidence in their completion of the task [25]. Moreover, trust in an automated system can depend on both individual differences in trust in automation generally [23] as well as experience with a specific system [128]. Understanding how different trust levels in users may impact their reliance on a model is critical to creating ML models that users will actually use.

The current study was motivated by the development of sensitivity analysis guided explanations that would be generated to help users understand how the features of a single data point contributed to the model's decision for that data point. The intended end-user group is a cybersecurity intrusion detection team. This team manually re-views ambiguously malicious emails that are flagged based on several security measures to determine whether they are indeed malicious. The goal of the explanation is to increase the efficiency and effectiveness of analysts by providing information about which features the ML model used to classify an email as malicious or benign.

As such, the current study investigates the decision-making impacts that different aspects of a ML explanation have on how users understand and use the outputs from ML explanation methods in a simulated spam detection task. We will operationalize explainability via task performance: if a model explanation improves a user's task performance relative to a baseline condition in which they did not receive an explanation, we will assume that the additional information provided by the model improved explainability. The current study examines the 1) impact of the presentation of the explanation and 2) characteristics of the ML model and user trust.

The presentation of the explanations examines whether the number of features and the visualization type of these feature importance values from the explanation of a machine-learned black box model impacted human decision-making. Specifically, we predict that if showing participants more features from the model improves performance, then we will see higher accuracy for both the three- and seven-feature conditions relative to baseline, and potentially higher accuracy for the seven- versus three-feature condition. We also predict that if visualizing the feature importance values graphically rather than numerically decreases cognitive load and makes it easier comprehend the explanations, then we will see higher accuracy in the graph versus numerical table visualization conditions.

The characteristics of the ML model and user trust examines the types of decisions made by the model (i.e., hit, false alarm) as well as the overall model accuracy (i.e., 50% overall accuracy vs 88% overall accuracy). This will allow us to investigate the interactions between visualization type, decision type, and model accuracy on user performance. Decision type and model accuracy have not been systematically investigated in the previous literature; controlling for these factors will enable us to extend our understanding of this potentially complex interplay. In conjunction, we will measure user trust in the model using the Trust in Automation scale [49] that was created specifically to measure trust in XAI applications. We will use this scale to assess whether behavioral differences are ob-served among individuals reporting different levels of model trust; for example, we would predict a higher level of compliance with model predictions for people with high trust versus low trust in the model.

6.1.1. Method

6.1.1.1. Participants

Participants were recruited through Amazon Mechanical Turk, following protocols approved by the Sandia Human Studies Board. Participants provided informed con-sent electronically by entering the current date. Data was collected from 222 participants. Data from three participants was excluded because they completed the study more than once; an additional 19 participants were excluded for missing more than one catch trial. The final dataset contained 200 participants. No demographic data was collected about participant's age or sex. However, in order to have a Mechanical Turk Account, individuals must be at least 18 years of age, and the researchers placed an additional restriction to only recruit individuals in the United States. Participants were paid $2.50 USD for participation in the study, based on pilot tests indicating that the task would take approximately 15 minutes or less to complete.

6.1.1.2. Materials

Email Stimuli. Stimuli consisted of 80 emails (40 spam and 40 not spam) gathered from several sources, including being drawn from the preexisting "Enron" dataset [7]. Thirteen binary features were created to describe the emails, based on similar features that appeared in other spam datasets as well as relevance to the current stimulus set and their ease of interpretation to end users (e.g., email contained spelling errors, urgency was implied). Binary yes/no values for each feature were manually assigned to each email. The full set of feature names is listed in Table 6-1 as well as the proportion of stimuli that had a "yes" value for each feature.

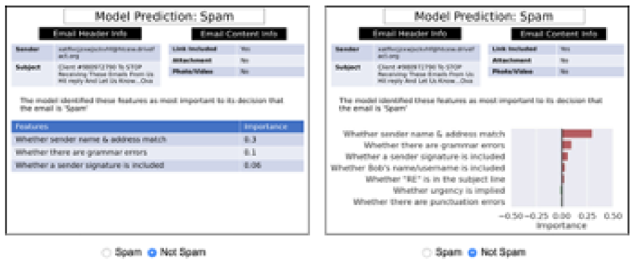
An initial set of 102 emails (50 spam, 52 benign) was used to train the classifier. A Random Forest ensemble classifier was trained on this derived dataset and the popular explainability library SHAP (based on game theoretic Shapley values) was utilized to generate realistic importance values for each feature in the model's classification decision for each email [69, 86]. The model achieved 97% accuracy, correctly classifying 48/50 spam emails and 51/52 benign emails. From this dataset, 40 spam and 40 not spam emails were chosen as experimental stimuli, on the condition that 1) the model accurately classified it, and 2) the sender and subject line could not be immediately recognized as spam (i.e., from "MARK ZUCKERBURG").

Table 6-1. Feature counts across the stimulus set. Features marked with * indicate that the feature was as a "stimulus information feature," and was presented on each trial regardless of experimental condition.

Feature Name	Percentage of Stimuli with a Positive Value on Feature	
	Spam	Not Spam
Email contains Link*	0.88	0.28
Email contains Attachment*	0.05	0.18
Email contains Photo*	0.48	0.30
Email Sender + Address match	0.20	1.00
Email contains Spelling errors	0.25	0.08
Email contains Grammatical errors	0.53	0.13
Email contains Punctuation errors	0.38	0.08
Email recipient name is mentioned	0.23	0.50
Email contains Symbols (e.g., Greek letters)	0.10	0.05
Email contains high count of "!"	0.08	0.08
Email contains high count of # sign	0.03	0.03
Email contains signature	0.13	0.30
Email urgency is implied	0.60	0.05

A base stimulus was created for each email, consisting of the Sender Name, Subject Line, and information regarding whether the email contained a Link, Attachment, and/or Photo (with a yes or no response for each; see Figure 6-1). This was to ensure that participants saw a minimal amount of information on each trial that could enable them to make a reasonable decision about the email without additional information from the model. All stimuli were shown with a prediction from the model (Spam or Not Spam), which was correct for 40/80 of trials in the 50% model condition, and 70/80 trials in the 88% model condition. On 80% of trials, participants also saw additional information from the model. We manipulated the number of features shown to participants from the model (0, 3, or 7). When model features were shown, we also manipulated the manner in which the feature importance values were visualized (graph or table). This resulted in ten visualization conditions (for each combination of model prediction accuracy, number of features, and visualization type). As such, ten versions of each email were created, with one level each of model decision accuracy (Correct, Incorrect), model feature number (0, 3, or 7), and, if features were shown, a feature importance visualization type (graph, table). Ten experimental lists were created to allow each stimulus to rotate through each of the conditions across participants. This ensured that idiosyncratic effects of individual stimuli were not confounded with experimental condition.

Within an experimental list, each participant saw 80 trials: 40 spam and 40 not spam emails. Within email type, each participant saw 8 trials per condition (i.e., 3 Features-Graph-Correct, 7 Features-Incorrect). Overall model accuracy was manipulated as a between-subjects factor. For participants in the 50% model accuracy condition, half of the trials within each visualization condition were shown with the correct classification and half with the incorrect classification. For spam emails, this means that participants saw a correct classification from the model for half of



The model identified these features as most important to its decision that the email is 'Spam'

Features	Importance
Whether sender name & address match	0.3
Whether there are grammar errors	0.1
Whether a sender signature is included	0.06

Whether sender name & address match
Whether there are grammar errors
Whether a sender signature is included
Whether Bob's name/username is included
Whether "RE" is in the subject line
Whether urgency is implied
Whether there are punctuation errors

Figure 6-1. Example stimuli used in experiment. The left panel is an example of the 3 feature table condition; the right panel is an example of the 7 feature graph condition. The "email header info" and "email content info" were displayed on every trial, as was a model prediction.

Table 6-2. Trust in Automation Scale

Question Number	Question Text
1	I am confident in the [tool]. I feel that it works well.
2	The outputs of the [tool] are very predictable.
3	The tool is very reliable. I can count on it to be correct all the time.
4	I feel safe that when I rely on the [tool] I will get the right answers.
5	The [tool] is efficient in that it works very quickly.
6	I am wary of the [tool].** [Reverse scored.]
7	The [tool] can perform the task better than a novice human user.
8	I like using the system for decision making.

the trials (i.e., Hit), and an incorrect classification for half (i.e., Miss). For benign emails, participants saw a correct classification form the model for half of the trials (i.e., Correct Rejection) and an incorrect classification form the model for half of the trials (i.e., False Alarm). In the 88% model accuracy condition, seven of the eight trials within each model visualization condition were shown with the correct classification, and one with the incorrect classification. Across both model accuracy conditions, incorrectly classified trials were shown with the same features and feature importance values; only the model's decision was reversed.

Additional Questions. The Trust in Explainable Artificial Intelligence Scale (Hoffman et al., 2018) was also collected to allow us to measure the impact of individual differences in trust on model reliance. This scale consists of eight questions/statements that individuals respond to with a 5-point Likert scale ranging from 1 (Strongly Disagree) to 5 (Strongly Agree). The questions are listed in Table 6-2. Numeric scores associated with the Likert ratings were summed (question 6 was reverse-scored) to generate a single composite score for each individual.

Each participant was also asked to answer three additional questions: 1) which visualization they found most helpful for decision making, 2) which visualization they found least helpful for decision making, and 3) how frequently they thought the model gave the correct prediction (Almost Never, About 25% of the Time, About 50% of the Time, About 75% of the Time, or Almost Always).

6.1.2. Procedure

Participants' task was to decide whether an email shown to them was Spam or Not Spam by clicking a radio button next to their preferred response (see Figure 6-1). After agreeing to the informed consent and clicking through the instruction screens, stimuli were presented in a random order. Each stimulus remained on the screen until the participant made a decision. Interspersed with the 80 experimental trials were eight catch trials, to gauge participant engagement. The catch trial overtly indicated which response the participant should choose (half required a Spam response, and half required a Not Spam response). If any participant missed more than one catch trial, their data was removed from analyses and they were replaced. Participants were offered a break screen every 22 trials, which were untimed and remained on the screen until the participant clicked a button to proceed. The average time from when the task was started to task submission was 46 minutes (min = 4 minutes, max = 176 min, SD = 44). Task completion times vary widely because participants were given several opportunities for untimed breaks, or could click away during any trial; as long as they submitted the task within three hours of accepting it they would receive credit.

6.1.3. Results

Data was cleaned in a two-step procedure. First, extremely long trials (e.g., 30 minutes) were excluded from analysis, as the indicated that the participant was not engaging in the task during those trials (e.g., they may have stepped away from their computer, given the online platform). Secondly, trials with response times longer than 3 SD above or below an individual's mean and/or longer than 20 seconds were discarded from analyses. In total, 240 trials were removed from analyses based on these criteria (70/200 participants had no trials excluded; 124/200 participants had between one and three trials excluded; 6/200 participants had between 4-5 trials excluded; no participants had more than 5 trials excluded from analyses).

6.1.3.1. Effects of Visualization

Our first question asked whether there were accuracy differences by visualization type, overall model accuracy, or their interaction. An analysis of variance (ANOVA) was conducted with the within-subjects factor of visualization type (5 levels: baseline, 3-graph, 3-table, 7-graph, 7-table) and the between-subjects factor of model accuracy (2 levels: 50% vs 88%). Mean values are listed in Table 6-3. The analysis found a significant main effect of model accuracy ($F(1,198) = 82.08$, $p < .001$). Overall participant accuracy was higher in the 88% model accuracy condition than in the 50% model accuracy condition (Figure 6-2). There were no effects of visualization type nor interaction between visualization type and model accuracy ($Fs < 1$, $ps > .53$). Our prediction that showing users additional model information would improve accuracy was not sup-ported by the data.

Next, we asked whether there were response time differences by visualization type, overall model accuracy, or their interaction (see Table 6-3). For this and all RT anal-yses, only trials that participants answered correctly were included in analyses. A similar ANOVA was conducted as

Table 6-3. Mean accuracy and response times by visualization type and model accuracy condition.

Model Accuracy	Visualization	Accuracy		Response Time (ms)	
Condition	Condition	Mean	SD	Mean	SD
50%	Baseline	0.57	0.11	3803	2187
	3 Features - Graph	0.57	0.13	4033	2511
	3 Features - Table	0.57	0.12	4071	2659
	7 Features - Graph	0.57	0.12	3994	2372
	7 Features - Table	0.58	0.13	4193	2733
88%	Baseline	0.73	0.18	3823	2348
	3 Features - Graph	0.74	0.18	4211	3035
	3 Features - Table	0.74	0.18	4024	2551
	7 Features - Graph	0.72	0.17	4158	3057
	7 Features - Table	0.74	0.18	4217	3163

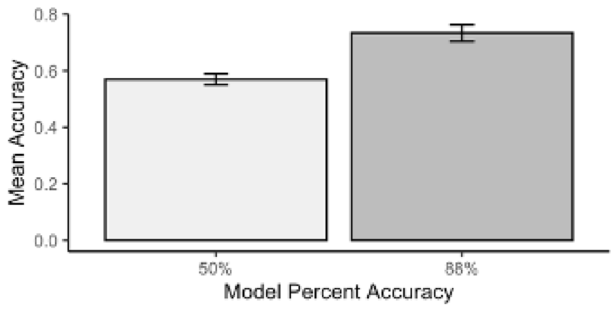


Figure 6-2. Mean accuracy for each of the model percent accuracy conditions, collapsing across visual-ization condition. Accuracy was significantly higher in the 88% than 50% model accuracy condition.

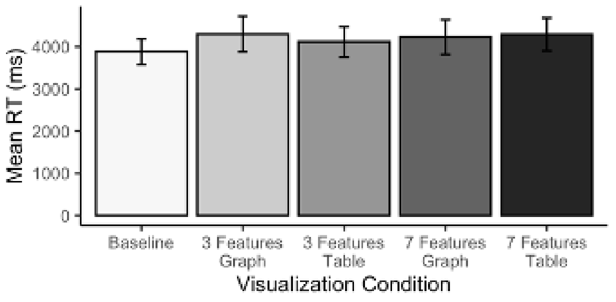


Figure 6-3. Mean RT for each visualization condition, collapsing across model percent accuracy. RTs were significantly longer in the 7 feature table condition than baseline.

described above, including only RTs for correct trials. The analysis found a significant main effect of visualization condition (F(4,792) = 2.74, p < .001), but no effect of model accuracy nor interaction between the two (Fs < 1, ps > .84). Follow-up pairwise comparisons between the visualization conditions, collapsing over model accuracy, found that the baseline condition was significantly faster than the 7-feature-table condition (Figure 6-3) when using the Bonferroni correction for multiple comparisons (t(199) = 2.95, p < .05). This is perhaps unsurprising, as the 7-feature-table condition showed users the most information, and so longer re-sponse times would indicate that they look longer to read the information. Our prediction that explanations presented in graphical format would be easier to read, and thus take less time, was not supported by the data.

6.1.3.2. Effects of Trial Type

Our second overall question was whether participant accuracy differed for different trial types (e.g., hit, miss), and whether this interacted with overall model accuracy (e.g., 50% versus 88%). Mean accuracy is shown in Figure 6-4. A repeated measures ANOVA was run with the within-subjects factor of Trial Type (4 levels: hit, miss, false alarm, correct rejection) and the between-subjects factor of Model Accuracy. There was a marginal effect of model accuracy (F(1,198) = 3.24, p = .07), a significant main effect of model output type (F(3,594) = 159.76, p < .001), and importantly, a significant interaction between the two (F(3,594) = 4.22, p < .01). Follow-up pairwise comparisons were conducted to explicitly test how participant accuracy within trial type differed based on the overall model accuracy. As seen in Figure 6-4, accuracy was numerically higher in the 50% model accuracy condition for the hits and correct rejections, whereas accuracy was higher in the 88% model accuracy condition for false alarms and misses. For false alarms, this difference in accuracy reached significance (t(198) = -3.22, p < .01). This effect could indicate that when the model was general-ly reliable and false positives were rare, people were better able to recognize this mod-el error and respond correctly than when this error was common.

Next we separated the data into visualization conditions to understand the effect of visualization type on the observed effects of trial type and overall model accuracy. We ran a mixed ANOVA, with the within-subjects factors of Visualization Condition and Trial Type, and the

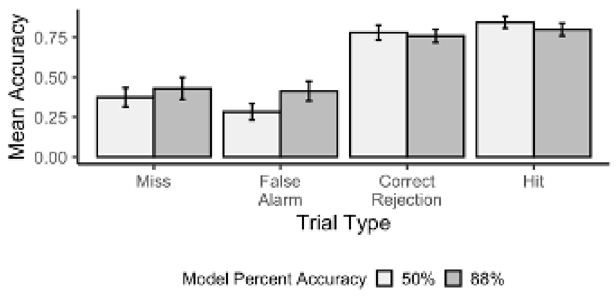


Figure 6-4. Mean accuracy by trial type and model percent accuracy, collapsing across visualization condition. Accuracy for false alarm trials was significantly higher in the 88% than 50% model accuracy condition; no other pairwise comparisons reached significance.

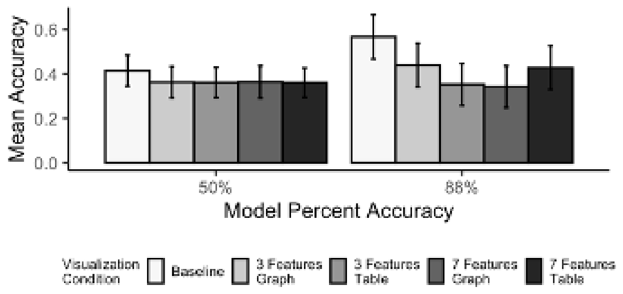


Figure 6-5. Mean accuracy for Miss trials only, by visualization condition and model percent accura-cy. In the 88% model accuracy condition, accuracy was significantly higher in baseline than all other conditions (except 3-features-graph).

between-subjects factor of Model Accuracy. This analysis showed a 3-way interaction (F(12,2196) = 2.20, p < .01), indicating that the effects of trial type and visualization type were different within the different model accuracy conditions. As such, we conducted follow-up tests to assess the effect of visualization condition within each level of model accuracy and trial type. This enabled us to understand how the model visualization type impacted accuracy for each trial type and accuracy level separately. One particularly interesting effect emerged, especially in light of the cybersecurity focus of the task, for which miss trials would carry high consequences. For miss trials in the 80% model accuracy condition (shown in Figure 6-5), we observed the highest accuracy in the baseline condition, with significantly lower accuracy for all other conditions (ts > 3.30, ps < .05) except the 3-table (t(84) = 2.53, p = .13). These findings suggest that when the model was overall “good” but missed a target, people were less likely to detect this error when the model provided more explanation information, relative to the no-explanation baseline.

6.1.3.3. Individual Differences in Model Trust

Average trust scores were calculated for each of the two model accuracy conditions, and interestingly they were almost identical across the two conditions (50% Model Accuracy: Mean =

Table 6-4. Likelihood of compliance with model's prediction by model trust score and overall model accuracy.

Trust Score Median Split	Model Percent Accuracy	Likelihood of compliance with the model's prediction	
		Mean	SD
High Trust	50% Accuracy	.76	.19
	88% Accuracy	.75	.19
Low Trust	50% Accuracy	.72	.17
	88% Accuracy	.76	.15

29.4, SD = 2.5; 88% Model Accuracy: mean=29.2, SD=3.0). We divided participants into High Trust and Low Trust groups based on a median split of trust scores (Median = 29) in order to examine group-level behaviors of people who reported different levels of trust in the model.

We looked at how trust and overall model accuracy interacted with whether people complied with the model's decision, and how quickly they did so. First, we examined the likelihood of model compliance for high versus low trust individuals in the 50% model accuracy versus 88% model accuracy conditions (see Table 6-4). Numerically, low trust individuals were more likely to comply with the model's prediction in the more accurate than less accurate model condition, whereas high trust individuals did not show this pattern. However, a between-subjects ANOVA with factors of Trust (2 levels: High Trust, Low Trust) and Model Percent Accuracy showed that neither main effect was significant, nor was there an interaction (all Fs < 1, ps > 0.35).

We also examined response times for trials on which individuals did or did not comply with the model's decision, across different levels of Trust but collapsing across model accuracy (see Figure 6-6). Thirteen individuals were removed from this analysis because they either always complied with the model's decision (10 participants) or never did (3 participants). A mixed ANOVA was conducted, with the within-subjects factor of Model Compliance (2 levels: Complied with model, Did not com-ply with model) and the between-subjects factor of Trust. Results showed a main effect of Trust (F(1,185) = 4.66, p < .05), and a main effect of Compliance (F(1,185) = 18.65, p < .01), but no interaction between the two (F(1,185) = 0.001, p = 0.99). These findings indicate that all participants were slower to respond when they did not com-ply with the model's prediction than when they did comply, and that low-trust individuals were generally slower to respond than high-trust individuals regardless of whether they complied with the model or not.

6.1.4. Discussion

This study investigated the impact of different types of ML explanations, the accuracy of a ML model, and a user trust in the ML model on user performance in a simulated spam detection task. Results showed that people performed the task with higher accuracy when they saw a more accurate model, but there was no overall effect nor interactions with feature number or feature importance visualization on accuracy. Response times were longer in the seven-feature table visualization than baseline, but this was likely driven by the additional time needed to read the extra information on the screen, as these increased response times did not correspond to an

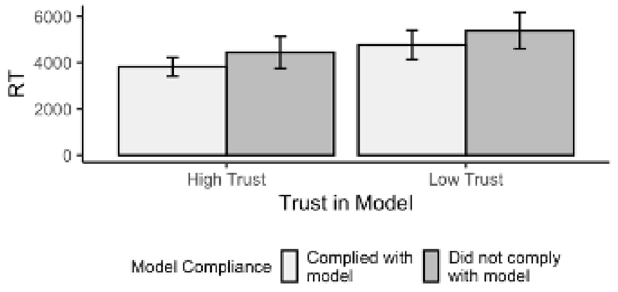


Figure 6-6. Mean RTs for trials in which people did and did not comply with the model's prediction by model trust median split.

accuracy difference. In general, our prediction that showing participants additional information regarding feature importance values would improve performance above and beyond the baseline condition, in which a model decision as provided with no explanation, was shown to be invalid. More interestingly, when considering different trial types (e.g., hits, false alarms), we found effects of overall model accuracy and interactions with visualization type in participants' ability to identify and overcome certain model errors. Participant accuracy for false alarms was better in the 88% accurate model condition than the 50% accurate condition. In other words, participants were better at correctly responding "Not Spam" to a false alarm made by the model in the 88% model accuracy condition, in which false alarms were rare. On the surface, this seems to run counter to some findings in the psychology literature that subjects will tend to miss rare events (Rich et al., 2008). However, it could be the case the participants implicitly learned the map-ping between the email header info and Spam/Not Spam categories across the course of the experiment, especially when the model was highly accurate, and thus created reliable schemas to help them correctly categorize these emails despite the incorrect model prediction. Future work should determine whether this effect is replicable and its cognitive underpinnings.

Moreover, within the 88% accurate model condition, participants were more likely to comply with incorrect "Not Spam" model predictions (i.e., miss trials) when the model provided more explanation information, compared to the no-explanation base-line. This is consistent with previous findings that users are more likely to rely on a model when more information is provided about its decisions [17, 60], regardless of whether that information is correct or not. Because this result was only observed in the highly accurate model condition, it raises risks regarding the implementation of such models in high consequence workflows. Caution should be exercised before as providing users with detailed ML explanations from accurate models, as it could overinflate compliance with model decisions, leading to missed targets.

Finally, we observed differences in behavior based on individual differences in model trust. Although all participants exhibited longer response times when they did not comply with the model's prediction, people reporting low trust in the model exhibiting longer response times than those with high trust regardless of whether they complied with the model's decision. This suggests that those with low model trust may have spent this additional time viewing the model's explanations and factoring that into their response choice. Future work should further investigate which aspects of ML explanations may engender higher or lower levels of trust, in order to help users develop appropriate trust and reliance on the system. Interestingly, we did not find differences in the likelihood of model compliance across users with high and low trust. This may indicate a strong bias to comply with a model's prediction, especially in an ambiguous task for which the user may have low confidence or little prior experience. Future work should investigate the factors that influence whether an individual complies with a model they believe to be incorrect, again in the service of helping users develop appropriate levels of trust and reliance on models deployed in high-consequence decision-making.

Our study is not without caveats. Because our data was collected online via Amazon Mechanical Turk, individuals in our study sample were likely novices with respect to both machine learning and cybersecurity. It is possible that findings may be different if we were to test an expert cybersecurity team, as they would have relevant domain knowledge to draw upon regarding the likelihood of certain email features being associated with malicious emails. There are aspects of our task that may have increased its difficulty or limited the usefulness of explanations, such as: limited context regarding the email (only saw features describing the email, not the actual text); forced binary choice (instead of ability to explore more features of the email to follow-up on model's prediction); or lack of feedback to users regarding task accuracy. It was also the case that our "incorrect" model decision trials used the same features and weights as in the correct instances, but just reversing the classification decision. Future work could train a model that actually misclassifies certain stimuli in order to produce more realistic feature weightings to use in misclassification decisions.

Our findings suggest that the efficacy of machine learning explanations for task completion depend, minimally, on the nature of the task, the overall model accuracy, the types of errors likely to be made by the model, and the user's trust in model. Future work should continue to explore how ML explanations impact human decision-making to ensure that the information provided does not hurt the human analysts' ability to perform high consequence tasks.

6.2. Lessons Learned from xAI Deployment in a Cybersecurity Operations Setting

Rapid improvements in artificial intelligence (AI) techniques have resulted in significant increases in their usage in a diverse and expanding set of applications. While original successes were in domains with fairly low consequences such as product and movie recommendations, AI algorithms are being used in increasingly higher-consequence applications such as medical diagnoses [28]. Wide-spread use is limited, however, as there is a recognized need to trust and understand the AI models before they are deployed and integrated into larger systems. In response, several explainable AI (xAI) techniques have emerged [4] to build trust and ensure that the models are fair.

Applying AI models in cybersecurity operations settings is a new and growing area, with strong emphasis on overcoming inefficiencies and uncertainties and improving overall incident response performance. Cyber-attacks result in loss of monetary resources and/or system resource availability. These attacks are increasing in volume and sophistication. AI methods offer improvement to the defense of cyber infrastructure, resulting in the preservation of significant resources. AI has been investigated in several cyber domains including malware detection [91] and malicious PDF detection [108]. xAI has been examined systematically using deep learning methods in cyber defense [122], but independent of the cybersecurity analyst.

We examined the use-case of AI models with explanations for identifying malware in a computer network defense setting. Attacks that occur on enterprise networks are of such a large scale that automated techniques are needed to help manage the attacks. Given the high impact of false negatives, cybersecurity analysts are highly skeptical of automated tools. To increase the productivity of the cybersecurity analysts, the AI model not only needs to be robust and reliable, but also the cybersecurity analyst needs to trust the model to make effective use of its output. However, AI and xAI methods are often deployed without evaluating how they affect the overt decision process.

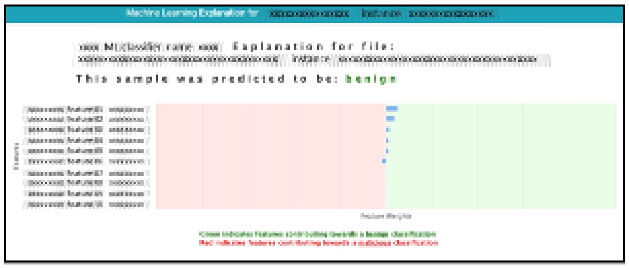
This chapter details a case study examining the usefulness of xAI techniques integrated into the workflow of cybersecurity analysts. In this setting, the cybersecurity analysts need to not only identify malicious artifacts, but also provide reasons why they are malicious. Hence, the goal of providing xAI methods is two-fold: to help scale with the increasing number of malicious attacks and to point to why the artifact is malicious as part of a cybersecurity analyst's workflow.

To assess human decision making when presented with model explanations, a user study was conducted with a broader population beyond cybersecurity analysts [112]. The study revealed that when making a decision about a potentially malicious stimulus participants often agreed with the xAI outputs, indicating high inherent trust in the model. We also found that the number of features presented was not a significant factor in the decision to agree with the model's recommendation or not.

The next step was to understand the use of xAI in a cybersecurity context with real analysts. To evaluate the effectiveness of the model and explanations, we planned to collect objective and subjective measures from actual end users in a live security setting. We planned to compare decision behaviors of analysts before and after the xAI deployment to determine if, and how much, cybersecurity analysts trusted and/or used the outputs.

When deploying these techniques in the real world, there were many decision points and limiting or confounding factors that had to be considered to conduct an evaluation of the new tool. This paper describes our evaluation and deployment of the xAI tool, as well as the lessons learned along the way about how cybersecurity analysts in general interact with xAI in real time. This paper does not focus on the visualization methods and design by which the xAI tool would display information to the human user. Rather, we present findings related to practical deployment of the tool. We also provide a list of considerations for AI developers and explanation designers that will help guide decisions during this process. Here we used TreeSHAP [68], but any xAI tool that provides feature importance for a prediction could be used.

Research Question: What practical considerations should be considered when developing and deploying AI and xAI tools in high-consequence, live settings?


Machine Learning Explanation for

ML Classifier name Explanation for file:

This sample was predicted to be: benign

Green indicates features contributing towards a benign classification

Red indicates features contributing towards a malicious classification

Figure 6-7. Obscured representation of the xAI tool output when expanded by analyst.

6.2.1. Methods

6.2.1.1. Evaluating Effectiveness of AI Tools

Cybersecurity analysts working in real-world incident response teams must make fast triage decisions using multiple pieces of information often including AI model outputs. In this use case, cybersecurity analysts triage multiple alerts to determine if flagged activity is actually malicious. Our goal was to evaluate the efficiency and effectiveness of a single AI model output in the context of incident handling before and after an xAI tool was introduced. We collected (1) instrumented data throughout each of two time periods: pre-xAI tool and post-xAI tool deployment, and (2) survey data from analysts after deploying the xAI tool. A visualization of the current xAI tool is shown in Figure 1.

The goal of the user study is to better understand the process used by cybersecurity analysts when promoting an alert and whether the analyst complied with AI model output. Data collection was programmed on the backend of existing cybersecurity tools to prevent the interruption of analyst workflow. Instrumented data collection included when/if an alert was promoted, model output, how the cybersecurity analyst interacted with the alert, and other activities performed on each alert. For the post time period, data indicating whether an analyst opened the xAI tool was also collected.

To understand the trust level of and satisfaction with explanations from xAI, end user perceptions were measured in a survey via two scales: the Trust Scale Recommended for xAI and the xAI Explanation Satisfaction Scale [48]. The Trust Scale measures whether end users are confident in the xAI tool, and whether the xAI tool is predictable, reliable, efficient, and believable. The Explanation Satisfaction Scale captures end users' judgments about the xAI tool. The cybersecurity analysts were invited to complete an online, 16-item questionnaire including these two scales after at least one week of working with the xAI tool.

6.2.1.2. AI Tools in a Live Security Setting

We identified some important attributes about the operational cybersecurity environment we studied to provide some context. First, there is a high cost of undetected malware. Intrusion detection systems are tuned to be sensitive because the cost of undetected malware can be extremely high. Second, intrusion detection systems include multiple, sometimes partially overlapping, alerting criteria. Third, within this context there is a bias towards hard cases; easily detectable malware is automatically mitigated with existing tools and, therefore, not triaged. Samples triaged by analysts are harder to classify and often involve contradictory predictions from competing (and highly accurate) mechanisms. Fourth, to automatically process files with AI algorithms, there is a semantic gap between real-world interpretation and low-level feature space for learning-based intrusion detection systems [106]. In other words, the interpretation of feature space is not self-apparent (such as is the case with some image classification problems) [110]. Notably, in the team we studied, the individuals who triage alerts are largely disjointed from individuals who maintain the AI models.

It was very challenging to collect data in a scientific way to assess the usefulness and efficacy of using xAI. This is a known challenge in cybersecurity operations settings [47], and we adopted knowledge already learned when constructing our hypotheses and initial research questions about the tool. However, as we devised the plan for collecting data towards answering those research questions, we discovered additional factors that refuted initial assumptions about how the tool would be used. Factors included:

Decision task and alternate decisions support paths. The analysts use the classification output from the AI model with other alert data to make a decision about an event; they may not regularly question the classification output. To mitigate this, we captured data from before/after the tool was deployed to see if including explanations changes analyst behaviors. We seek to capture measurements such as if the explanations are expanded and average response time. We also made the visual presentation of explainability more palatable compared to previous versions, which did not organize or present explanations in ways that could be quickly utilized during the decision-making process.

Workflow. Much of the information cybersecurity analysts use to make a triage decision exists in a central incident handling tool, with little navigation required within the dashboard to find decision-critical information. This is a standard workflow in cybersecurity operations settings, and thus perhaps less critical for our own studies. However, user workflow should be considered prior to deployment of xAI techniques in some fashion to understand potential friction points for adoptability.

Tool separation/location. The xAI tool exists outside the main dashboard where analyst conclusions are registered; it is located in a supporting software program which requires pivoting to engage with it. While this program is routinely accessed by analysts, the addition of the tool was not immediately obvious. To mitigate this, we (1) hosted training with the analysts so they would be able to locate the xAI tool, and (2) created an interface feature (Figure 1) to increase salience of the new xAI tool.

Number of end users and their roles. In our scenario, there is an assigned primary incident responder per week causing turnover and rotation within the group of users whose roles differ regarding decision-making about an event. To mitigate this, we include all users who interact with the explainability tool, not just the incident responders who are primarily responsible for incidents in a given week.

6.2.2. Results

We collected instrumented data without interrupting the incident handling context with and without xAI tools. We then measured user trust and perception of usefulness of the xAI tool.

As described in Section 6.2.1, quantitative data were collected continuously over the course of several months; we monitored this data stream to capture a pre-deployment baseline of existing tool use and post-deployment data to ensure the xAI tool was working properly. Data were collected for 36 days pre-tool implementation and 43 days post-tool implementation. A total of 2834 unique alert IDs that were triggered by the macro or pdf_url classifier were included in the data. Of the 2834 alerts, 17 were promoted to events which are the focus of this analysis (Table 6-5).

One hypothesis we had was that the availability of a novel explainability tool would change the likelihood of an analyst seeking out an explanation from ML models. Surprisingly, we discovered that users were not interacting much with the existing tool or the new tool. As shown in Table 6-5, out of 9 events that occurred pre-explainability tool, the pre-existing explanation tool was opened 2 times (22%). Of 8 events post-explainability tool, the new explanation tool was opened 2 times (25%). Why were these analysts not often opening the explanations, even prior to the implementation of a novel explainability tool? A shift in thinking allowed us to appreciate the key finding in the pre- deployment data: the targeted analysts did not use explanations in their daily workflows. Moreover, the placement of a new tool in an inconspicuous location will further decrease the likelihood that users engage with the tool and relying on training to overcome that limitation is an insufficient strategy.

Another hypothesis was that the introduction of the explainability tool would change the length of time it took for an analyst to make a decision and close the event. When investigating this hypothesis we considered only the events where the explainability tool was opened and found that the average length of time (in minutes) for the 2 events pre-explainability tool (where the existing tool was used) was 85 minutes, whereas for the 2 events post-implementation the average time was 724 minutes. Note that one event post-tool was open for more than one day (Table 6-6).

We thought that an analyst's rate of compliance with the model output might change depending on whether they were using their old tool, or the new tool to view the model explanation. In this case, of the four events (pre- and post- tool period), the analysts agreed with the classifier output and in the fourth instance the classifier's output was uncertain (Table 6-7). We could not detect a difference pre- and post-tool.

Table 6-5. Data collected from 17 SCOT entries. LB=LaikaBoss

Period	Opened Explanation in LB	Classifier Prediction	Analyst Agreed with Classifier	Time Event Open (minutes)	(notes)	Total SCOT Event Views	Analysts with SCOT Enttries Count
Pre-Tool	N	malicious	Y	0.50		19	1
Pre-Tool	N	benign	Y	30554.9	21 days	31	1
Pre-Tool	N	benign	Y	39.9		22	1
Pre-Tool	N	benign	Y	11.7		21	1
Pre-Tool	N	benign	Y	217.8		27	0
Pre-Tool	N	malicious	Y	35.5		21	1
Pre-Tool	Y	benign	Y	120.7		60	2
Pre-Tool	N	benign	Y	9.9		32	0
Pre-Tool	Y	malicious	Y	48.3		18	0
Post-Tool	Y	uncertain	N/A	52.0		43	1
Post-Tool	N	malicious	Y	0.87		11	0
Post-Tool	N	benign	Y	5.55		10	0
Post-Tool	N/A	benign	Y	8.3	LB was not opened	77	1
Post-Tool	N	benign	N	N/A	Still under investigation/no close time	85	1
Post-Tool	Y	benign	Y	1395.7	1 day	27	2
Post-Tool	N	benign	Y	7.55		16	1
Post-Tool	N	malicious	Y	79.0		26	2

Table 6-6. Time to close events where explanations were viewed.

Period	Time Event Open (minutes)	Average Time Event Open
Pre-Tool	120.7	85
Pre-Tool	48.3	
Post-Tool	52.0	724
Post-Tool	1395.7	

Table 6-7. Analyst agreement with classifier in events where explanations were viewed.

Period	Classifier Prediction	Analyst Agreed with Classifier?
Pre-Tool	benign	Y
Pre-Tool	malicious	Y
Post-Tool	uncertain	N/A
Post-Tool	benign	Y

Table 6-8. Unique analysts with SCOT entries for events where explanations were viewed.

Period	Total SCOT Event Views	Analysts with SCOT Entries Count	Average Unique Analysts with SCOT Entries
Pre-Tool	60	2	2
Pre-Tool	18	0	
Post-Tool	43	1	1.5 (2)
Post-Tool	27	2	

When thinking about an event, multiple analysts can add information into the event log via the SCOT tool. We wondered if events where an explanation was viewed would have a different number of unique analyst entries. In Table 6-8, we show that there is no difference.

Finally, we thought that analysts might more often view an event if an explanation was opened at all. We found very little difference in the number of SCOT event views during the pre-tool phase versus the post-tool phase (Table 6-9).

Also as described in Section 6.2.1, qualitative data were collected in the form of an online survey sent to N=11 analysts. Questions probed user trust and perception of usefulness of the xAI tool [49] (questions listed in Table 6-2), one analyst responded to the request, are listed in table 6-10. The one respondent disagreed strongly with the statement, "This explanation of the model shows me how accurate the model is", and agreed strongly with, "From the explanation, I understand how well the model works." Due to small sample size we were unable to analyze our instrumented event data by an average trust score.

Table 6-9. Number of total SCOT views for events where explanations were viewed

Period	Total SCOT Event Views	Average SCOT Event Views
Pre-Tool	60	39
Pre-Tool	18	
Post-Tool	43	35
Post-Tool	27	

Table 6-10. Survey data from one analyst (of 11 possible, response rate = 9%), $n = 1$.

Question	Response	Numerical Rating
Q.1.1 I am confident in the explainability tool. I feel that it works well.	I agree somewhat	4
Q.1.2 The outputs of the explainability tool are very predictable.	I'm neutral about it	3
Q.1.3 The tool is very reliable. I can count on it to be correct all the time.	I'm neutral about it	3
Q.1.4 I feel safe that when I rely on the explainability tool I will get the right answers.	I agree somewhat	4
Q.1.5 The explainability tool is efficient in that it works very quickly.	I agree somewhat	4
Q.1.6 I am wary of the explainability tool.	I'm neutral about it	3
Q.1.7 The explainability tool can perform on the task better than a novice human user.	I agree somewhat	4
Q.1.8 I like using the system for decision making.	I agree somewhat	4
Q.2.1 From the explanation, I understand how the model works.	I agree strongly	5
Q.2.2 This explanation of how the model works is satisfying.	I'm neutral about it	3
Q.2.3 This explanation of how the model works has sufficient detail.	I agree somewhat	4
Q.2.4 This explanation of how the model works seems complete.	I disagree somewhat	2
Q.2.5 This explanation of how the model works tells me how to use it.	I disagree somewhat	2
Q.2.6 This explanation of how the model works is useful to my goals.	I agree somewhat	4
Q.2.7 This explanation of the model shows me how accurate the model is.	I disagree strongly	1
Q.2.8 This explanation lets me judge when I should trust and not trust the model.	I disagree somewhat	2

6.2.3. Discussion

Despite our efforts to understand analyst interaction with the system using unobtrusive data collection, we faced several challenges in deploying the tool in a live setting. Due to the chosen location of the tool, we expected some level of low engagement. To mitigate this risk of low familiarity with the tool's existence, we conducted a single-day training, which covered an overview on the tool's user interface, as well as a tutorial on its operational use. However, not all analysts were able to attend the training, and some analysts identified this as the reason they did not use the tool.

The explanation capability was added to existing intrusion detection systems with the assumption that understanding model rationale would help with triage tasks. One of the core insights gained is that this is a false premise. Taking the time to understand the rationale of one of many possible, and often contradictory, detection mechanisms is not necessarily the most efficient path for triage. This is especially true when analysts have other sources of information available to them that are more easily consumable, including the observation itself (e.g., the file that might be malware) and data views that have been developed based on analyst feedback. To some degree, performing the same manual analytic steps on samples regardless of alert source might ensure consistency and help prevent analytical bias.

The xAI tool targeted analysts based on the hypothesis that improved understanding of the model's decisions would increase analyst confidence and improve overall performance. Due to widespread skepticism amongst security analysts, this hypothesis made sense: provide more data such that their skepticism is satisfied. However, we believe that injecting a new xAI tool over existing models that analysts already trusted impact our ability to detect gain in confidence and reduction of skepticism.

We were also prepared to face challenges related to the environment in which the xAI tool was deployed. Though we found this to be not as relevant for our use case, the context of the xAI deployment may impact its usefulness and adoptability. Security incident responders are known to experience high load of alerts, and are subject to different kinds of cognitive biases when interacting with intrusion detection systems [64]. These settings have a history of high turnover and burnout [12, 95], and judgments about alerts are often made with pressure from long queue of alerts or time expected to make a decision [89]. More relevant to our use case was that the actual workflow of incident response analysts did not include validation of detection mechanism outputs (xAI or not), and rationalizing those outputs is not an efficient path.

Our own efforts resulted in some lessons learned for deploying AI models "in the wild", which might be useful for others developing xAI for use in real-world settings. The study originally aimed to conduct more controlled field experiments, which quickly evolved into tool improvement. Over the period of about 6 months, we were forced to modify our original study design to the extent that we developed new research questions and pursued entirely new studies. For some researchers, these changes represent some level of risk, which we believe can be mitigated by learning from studies like ours and considering certain design and deployment elements before commencing data collection.

Table 6-11. Practical considerations for xAI deployment

Practical Considerations	Supporting Questions
Who are your end-users?	Who uses the model outputs, and in what way?
	How does the xAI tool help them accomplish their goals?
	With respect to explainability, who critically questions how the model works (within their normal workflow)?
What is the context in which the model is deployed?	Do environmental pressures counteract the availability of the model?
	Are the features, feature names, and visual representations of explainability relevant and meaningful in this context?
What is the relative risk of the model being wrong?	How does the risk of model inaccuracy impact the end user?
	What are the consequences of trusting the model?
What is the risk of the explanation being unclear or incorrect?	How does an unclear explanation impact the end user?
	What are the consequences of presenting a poor or incorrect explanation?

We learned that our user base seemingly trusts the output of the AI model to the extent that they do not explore provided novel explainability tools, similar to previous conclusions on non-expert users [11]. Further investigation revealed that incident responders, or the people who are making decisions from the AI outputs (and a suite of other tools), are not interested in validating the model. However, this realization led us to consider two new questions: (1) who would validate the model outputs, and thus potentially benefit from xAI tools, and (2) how can the incident responders still contribute to the quality of AI explanations?

We found that AI model maintainers, or the experts tasked with training AI models and monitoring their performance, are more invested in verifying model outputs for the purpose of improving model accuracy. Accordingly, we have pivoted our efforts to understand that user base better. Future research also includes exploring the second question of increasing input from incident responders into AI tools without interrupting normal workflow.

We also learned that the context in which the xAI tool was deployed dictated how much it might actually be used and for what purposes. Though most of this lesson was learned through literature review, we still found that some aspects of context impacted how we deployed the xAI tool. For instance, we considered how the presentation of information in the explanations could be improved such that the outputs were meaningful to the target users. Additional contextual factors, such as time pressure, task volume, and consequences of trusting xAI tools, were found to be less critical for our use case but should be considered for researchers and developers planning to deploy such tools in real environments. Based on the above lessons learned, we offer considerations for developing and deploying xAI tools in live contexts. The questions in Table 6-11 can help guide decisions and mitigate risks during various stages of technology transfer.

6.2.4. Conclusions

While conducting a study aimed to understand if cybersecurity analysts would benefit from an xAI tool in a real-world setting, we learned that a team of cybersecurity analysts seemingly trust the output of the AI model and do not explore the provided explanations. Rather, other existing tools are used to validate the output of AI models. In this context, the use of the output from the AI classification model was embedded in cybersecurity analysts' main workflow, while the use of the new xAI tool was not.

We identified considerations that researchers and developers can integrate into current processes to design and target better xAI tools for more successful technology transfer. Additionally, examining real-time, nonintrusive data from instrumented backend data collection is a great means to understand if and how end users are using an xAI tool. Ultimately, considering the end users and their context early in the process reduces risks and impact of unidentified challenges.

6.3. Explainability for Model Maintainers

6.3.1. Objective and Research Question

The user study conducted in the cybersecurity operations setting revealed several facts about MLE, one of which was that the incident handlers are one of at least two (2) possible user groups. The second user group is comprised of model maintainers who are responsible for building, training, and maintaining (or updating) machine learning (ML) models deployed in the incident handling domain of cybersecurity operations. To better understand this user group and capture new potential requirements for the explainability tool, we conducted a third, short study using nonexperimental exploratory methods. Our research question for this study was:

How do model maintainers (and related supporting roles) interact with the model and explanation?

All of the examples relate to using the open-sourced PDFrate dataset, but are applicable to specific models used within a defined workflow.

6.3.2. Method

The population of model maintainers is even smaller than the incident handling group with a total of three (3) persons who could conduct this job. However, as our findings reveal, the roles of these individuals vary greatly. To capture a wide variety of potential factors, we conducted one-hour semi-structured interviews with the model maintainers (n = 3) following a set of questions developed to understand roles and goals of each participant. The interview protocol is included in Appendix E.

The qualitative interview data was analyzed by one researcher with experience in qualitative coding and analysis. Statements from the participants were identified by thematic interest and summarized in the findings. Due to low sample size and high variation across the sample, we present this research as useful insights for current and future work as it pertains to ML models in practice. However, our research findings should not be interpreted as generalizable beyond the operational environment from which they were collected.

6.3.3. Findings

6.3.3.1. Roles

Each of the participants had different roles when interacting with an ML model of interest.

The first two participants had supporting roles that aid incident handlers when needed, improve the model through identification of specific samples, and help monitor the model's performance. In addition to this support, both participants had unique roles supporting at least one additional facet. The first participant had deep knowledge of underlying models and had a major role in creating the ML model. This knowledge helps this person identify specific samples for model maintainers to consider, ensuring coverage of new/emerging threats. The second participant had some knowledge of underlying models and fills an analyst role in addition to supporting model maintainers. This unique role is to try to make the model more usable to analysts. This person's prior experience as an analyst helps in understanding how ML inference can be used to make decisions for specific observations.

The main model maintainer role is held by a single individual. The scope of their work includes developing classifier algorithms, identifying new features, building a process for training and testing, deploying a ML model, and model maintenance at a regular interval (weekly/monthly).

Model maintenance and retraining is continuous, and likely indeterminant in length as the model maintainer continues to add new samples to the training set. The process happens in 3 phases. First, the maintainer must find incorrect predictions and edge cases, conducted through a case by case review. Cases can be identified by supporting roles and analysts or through the maintainer's own queries and analysis. Next, the model maintainer must judge if particular samples are benign or malicious, sometimes redoing incident analysis to reach a conclusion. Like the analysts, the model maintainer relies on contextual information about the observation/case. This contextual information is extraneous to the model itself; this could include the actual artifact (email, pdf, etc.) and summary data from Splunk, but varies by case. Last, the model maintainer must decide if the case is relevant to improving the model by adding to the training set.

6.3.3.2. Model Goal/Health Metrics

The goal of the ML model, and thus of the model maintainers, is to detect threats reliably and consistently. Complete detection is not realistic given the dynamic, ever-changing nature of cyber threats and the nature of the model as a random forest model. Despite this unknown, there are some quantifiable metrics that are used to help gauge model performance. Namely, uncertainty and incorrect classifications should trend to zero. Overall, the expectation is incremental improvement over the long-term. This is achieved by adding new observations that train the model to detect new and unique threats.

The model itself is meant to assist a human in making a judgment about a given observation. Thus, two *effectiveness* metrics are linked to analyst interaction. The model should indicate when results are uncertain to draw human attention, and the tool should monitor how often analysts look at the explanations as a potential indicator of usefulness.

6.3.3.3. Key Aspects of the Model and its Outputs

The following points are key aspects of the model such that it can meet the decision-making needs of both analysts and model maintainers. Note that these model characteristics and outputs apply to both individual observations (as seen in the analyst workflow) and at the overall model level (as seen in the model maintainer role).

- Confidence / Certainty of prediction: this is perhaps the most important piece of information beyond the prediction itself for the analyst and model maintainer to judge the output of the model. All participants noted this was missing from the current explainability tool, and that it would be difficult to critically evaluate the model output without it.
- Classification accuracy: Was the model correct? Accuracy requires knowledge of ground truth (currently determined by a human retroactively) but could be a later addition by the human user to help evaluate model performance.
- Feature filters: The top 10 features only show a small portion of the values that contribute to the overall prediction. It may be helpful to have the option to see all features (or at least know the total number of features used in a given instance) or a user-constrained set of features such that they have appropriate framing for the judgment.
- Total values for benign/malicious/overall: The Top 10 shortlist of features can be somewhat confusing. Analysts are expected to evaluate the prediction against the numeric outputs of the model, but due to the number of features it is unrealistic to show all features and feature values. Alternatively, the model could show the total values for benign and malicious-predicting features, as well as the net value, to share with the analyst how "close" the prediction was to the center. The total magnitude in either direction would help analysts understand how close the prediction was to uncertain.
- Feature-specific information:
  - Feature definitions: Currently the feature names are somewhat obscure if the user does not have knowledge of the model's architecture and function. For instance, the feature name “email_to_domain_other” is somewhat easy to parse as “domain” is either internal or external, whereas “pdf_text_keyword_view” is more challenging to decipher with multiple potential meanings.Model maintainers have this knowledge, but analysts may not. It is important to give this meaning to a user such that they can properly ingest the information and make judgments.
  - Raw feature values (pre-normalization): These data would indicate a feature-specific number, which may not be useful in all contexts. Not all users said this would be important, especially if the user is not familiar with the distribution of each feature.

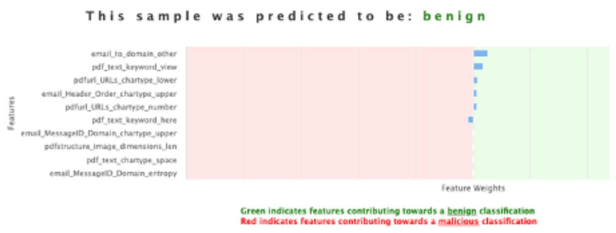


Figure 6-8. Current explainability visualization with benign and malicious feature weights does not allow the user to manipulate the outputs in any way. Giving the analyst some control over the analysis interface will help them filter to information they think is relevant or better understand the context of the outputs for more effective decision making and perception of usefulness.

- Sparkline of the distribution: This small visual would show where the observation falls in the distribution for that feature. This addition could complement the raw values as contextual information to help decision making.

- Global feature importance: This would be the same set of numerical values for the set of features included in the model. Global feature importance measures the importance of the feature for the entire model. It indicates how much impact that one feature, out of hundreds, has on a classification outcome. Building on the previous example, perhaps the feature checking whether email domain is internal versus external would be significant contributor to a classification outcome in this model checking for malicious emails. The global feature importance would not change observation-to-observation unless, or until, the model is retrained.

- Class feature importance: This is different for each observation. Local feature importance measures the contribution of the feature for that specific observation. For instance, observation K was run through the model, and observation K is an email that has an image included. In the case of observation K, in this one instance it is possible for “pdfstructure_image_dimensions_len" to be a bigger contributor to the prediction (model outcome) than average/other observations. The set of numerical values for the set of features used in this observation would vary.

- Information that triggered the feature value (context): For instance, if a particular feature strongly contributes to the prediction of “malicious", the feature itself might not give enough information for the analyst to judge the outcome. The analyst may want to know what exactly caused the feature to produce its result; this information is not currently included in or accessible through the explainability tool. Currently, analysts must pivot to the actual observation and its artifacts (e.g. email, attachments, etc) to find this information. Reducing the need to pivot between tools would save the analyst time while also increasing confidence in the tool.

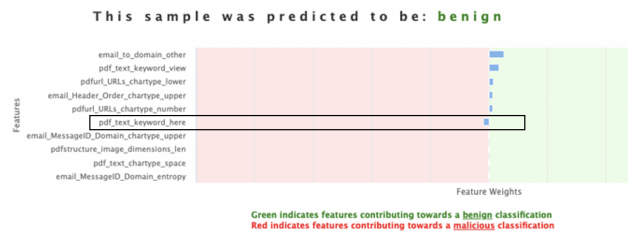


Figure 6-9. Explainability visualization shows "pdf_text_keyword_here" as a feature contributing to "malicious". However, it is unclear whether the high value is due to a large number of the word "here" in the pdf text, or due to a small number of the word "here" in the pdf text.

6.3.4. Discussion

6.3.4.1. Use Cases for Explainability Tool

Based on statements from participants, the following examples demonstrate how the ML model explainability tool might be used in both the analyst and model maintainer contexts.

1. Evaluating residuals: Being able to see and explore the residuals would help in determining if new features are needed.

2. Hypothesizing new features: Being able to see the model from a higher level; what features are already included and their respective coverage would help in hypothesizing if/which features should be added to the model.

3. Finding similar examples: It is helpful to be able to evaluate an observation against similar records. Helping an analyst or model maintainer identify those observations and pivot from the tool would help improve this comparison process.

4. Specialized/custom queries: Analysts and especially model maintainers expressed interest in being able to view multiple observations and control those outputs using specialized or custom queries.

5. Investigate specific aspects of an observation: It would be helpful to identify an aspect of the explanation to draw the analysts' attention to something specific in the email/pdf. For instance, if a feature indicates that there is something present (or missing) in the email or pdf that indicates a potentially malicious artifact, providing that information (what is the actual evidence observed by the model) to the analyst would improve their confidence and efficiency in evaluating the classifier output.

6.3.4.2. Other Findings

One finding of note was that participants from user studies 1, 2, and 3 indicated a trust dynamic that might not match how we are thinking about the problem. The main person who performs model maintenance indicated more trust in the analyst's decision than the model, but User Studies

1 and 2 indicated that people in general trust ML predictions and analysts are likely to agree with model outputs (respectively).

We found that the explainability interface likely has low generalizability to other models and processes within Security Operations. Other ML models are employed, but they are commercial models embedded in purchased tools and not within the control of model maintainers to train and prune them.

Practically speaking, there are a lot of (raw and meta) data associated with events that analysts evaluate and we noted some challenges in the evaluation and training processes that could be addressed without ML or explainability. For instance, when viewing an event flagged by a classifier, analysts are interested in identifying what portions of a PDF caused the classification. However, this is not immediately obvious, and the existing tool/platform does not tie back to the original PDF, but rather simply provides a list of features. Moreover, features in the model that contribute to "malicious" do not provide evidence in the same window, and the analyst is forced to pivot in order to properly evaluate the feature and corresponding information that triggered it. These small improvements in usability would help by improving confidence and efficiency of incident response analysts.

7. CONCLUSIONS

In retrospect, this LDRD was quite ambitious and could have been carved out into separate research projects of their own. Despite being quite ambitious, several future research veins are exposed that could be pursued:

- **Extending the application of SA and other mathematical analysis to ML.** SAGE was able to expose several key underlying issues that limit the effectiveness of explainability methods outlined in this report. A direct path forward would be address these issues. We also only examined SA, but there are several other possibly mathematical frameworks that would be pursued.
- **Quantifying the trust of an ML model prediction.** SAGE initially developed the GeoTrust metrics and made a connection to a relatively small effort in the ML community that researches the robustness of nearest-neighbor type methods. These methods by necessity depend upon the geometry of the training data. However, the curse of dimensionality severely limits its applicability. A direct path forward here could investigate how to expand beyond small feature spaces. This would also help the application of mathematical frameworks which are also limited by exploding computational constraints as the number of features increases. Another direction forward could be to tie the GeoTrust metrics into the existing methods that directly or indirectly leverage nearest-neighbors such as DeppKNN [84] or Radial Basis Functions [129]. We note that from our experiments in Chapter 2, nearest-neighbor approaches are not able to capture correlations in the feature space as well as other ML algorithms.
- **Quantifying and optimizing the impact explanations on end users.** The intention of SAGE was to measure directly the impact of explanations on end-users in an established workflow. However, we mostly discovered the overall ineffectiveness of explanations. There are several research questions including the most obvious of how to improve the effectiveness of the explanations or what would constitute an effective explanation

In addition to these direct research directions, recently there has been interested in explanations and adversarial vulnerability. The mathematical framework established as part of SAGE could be used to help study this intersection as well as privacy preserving ML. We conclude that SAGE has investigated a very timely and important issue to national security as explainability is desired by many customers because of the high-consequence applications. However, often the explainability methods are simply provided *without* understanding the impact of the explanation on the end-user and possible information could be leaked (i.e. possibly measured using Fidelity). We expect that these issues will be high-interest topics for research going forward and hope that SAGE has helped to facilitate future high-impact research.

REFERENCES

[1] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *arXiv preprint arXiv:1903.10464*, 2019.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in Neural Information Processing Systems*, 31:9505–9515, 2018.

[3] Algorithmia. Machine learning in production: a roadmap for success. Technical Report MSU-CSE-06-2, Algorithmia, 2020.

[4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[5] S. R. Arwade. Translation vectors with non-identically distributed components. *Probabilistic Engineering Mechanics*, 20:158–167, 2005.

[6] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[7] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[8] David Avis, David Bremner, and Raimund Seidel. How good are convex hull algorithms? *Computational Geometry*, 7(5):265–301, 1997.

[9] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] Markus Borg, Cristofer Englund, Krzysztof Wnuk, Boris Duran, Christoffer Levandowski, Shenjian Gao, Yanwen Tan, Henrik Kaijser, Henrik Lönn, and Jonas Törnqvist. Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. *arXiv preprint arXiv:1812.05389*, 2018.

[12] J P Bourget. Addressing analyst fatigue in the soc. `https://www.brighttalk.com/webcast/288/224207`, 2016.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[14] Vanessa Buhrmester, David Münch, and Michael Arens. Analysis of explainers of black box deep neural networks for computer vision: A survey. *arXiv preprint arXiv:1911.12116*, 2019.

[15] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K Varshney, and Dawn Song. Anomalous example detection in deep learning: A survey. *IEEE Access*, 8:132330–132347, 2020.

[16] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[17] Adrian Bussone, Simone Stumpf, and Dympna O'Sullivan. The role of explanations on trust and reliance in clinical decision support systems. In *2015 international conference on healthcare informatics*, pages 160–169. IEEE, 2015.

[18] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[19] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[20] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[21] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Approximate convex hulls family for one-class classification. In Carlo Sansone, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 106–115, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[22] Arjun Chandrasekaran, Viraj Prabhu, Deshraj Yadav, Prithvijit Chattopadhyay, and Devi Parikh. Do explanations make vqa models more predictable to a human? *arXiv preprint arXiv:1810.12366*, 2018.

[23] Jason A Colquitt, Brent A Scott, and Jeffery A LePine. Trust, trustworthiness, and trust propensity: a meta-analytic test of their unique relationships with risk taking and job performance. *Journal of applied psychology*, 92(4):909, 2007.

[24] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

[25] Peter De Vries, Cees Midden, and Don Bouwhuis. The effects of errors on system trust, self-confidence, and the allocation of control in route planning. *International Journal of Human-Computer Studies*, 58(6):719–735, 2003.

[26] Sukanta Dey, Abhishek Kumar, Mehul Sawarkar, Pranav Kumar Singh, and Sukumar Nandi. Evadepdf: Towards evading machine learning based pdf malware classifiers. In *International Conference on Security & Privacy*, pages 140–150. Springer, 2019.

[27] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[28] Bradley J Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L Kline. Machine learning for medical imaging. *Radiographics*, 37(2):505–515, 2017.

[29] Hubert Etienne. When AI ethics goes astray: A case study of autonomous vehicles. *Social Science Computer Review*, 2020.

[30] Shi Feng and Jordan Boyd-Graber. What can ai do for me? evaluating machine learning interpretations in cooperative play. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 229–239, 2019.

[31] Richard V Field, Jr. Stochastic models: theory and simulation. (SAND2008-1365), 3 2008.

[32] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. Model class reliance: Variable importance measures for any machine learning model class, from the "rashomon" perspective. *arXiv preprint arXiv:1801.01489*, 68, 2018.

[33] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[34] Edwin Ernest Ghiselli. *Theory of psychological measurement*. McGraw-Hill, 1964.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[36] Ian Goodfellow and Nicolas Papernot. The challenge of verification and testing of machine learning. *Cleverhans-blog*, 2017.

[37] Ben Green and Yiling Chen. Disparate interactions: An algorithm-in-the-loop analysis of fairness in risk assessments. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 90–99, 2019.

[38] M. Grigoriu. *Applied Non-Gaussian Processes*. P T R Prentice-Hall, Englewood Cliffs, NJ, 1995.

[39] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1321–1330. PMLR, 06–11 Aug 2017.

[40] Mark A Hall. Correlation-based feature selection of discrete and numeric class machine learning. 2000.

[41] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.

[42] Joseph Hart and Pierre A Gremaud. An approximation theoretic perspective of sobol'indices with dependent variables. *International Journal for Uncertainty Quantification*, 8(6), 2018.

[43] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2018.

[44] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *Proceedings of the International Conference on Learning Representations*, 2019.

[45] Bernease Herman. The promise and peril of human evaluation for model interpretability. *arXiv preprint arXiv:1711.07414*, page 8, 2017.

[46] Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human factors*, 57(3):407–434, 2015.

[47] Robert R Hoffman. The concept of a "campaign of experimentation" for cyber operations. *The Cyber Defense Review*, 4(1):75–84, 2019.

[48] Robert R Hoffman, Gary Klein, and Shane T Mueller. Explaining explanation for "explainable ai". In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 62, pages 197–201. SAGE Publications Sage CA: Los Angeles, CA, 2018.

[49] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

[50] Giles Hooker and Lucas Mentch. Please stop permuting features: An explanation and alternatives. *arXiv preprint arXiv:1905.03151*, 2019.

[51] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[52] Bertrand Iooss and Clémentine Prieur. Shapley effects for sensitivity analysis with correlated inputs: Comparisons with sobol' indices, numerical estimation and applications. *International Journal for Uncertainty Quantification*, 9(5), 2019.

[53] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning attacks. *arXiv preprint arXiv:2012.03765*, 2020.

[54] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

[55] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992.

[56] Ronald T Kneusel and Michael C Mozer. Improving human-machine cooperative visual search with soft highlighting. *ACM Transactions on Applied Perception (TAP)*, 15(1):1–21, 2017.

[57] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996.

[58] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with relieff. *Applied Intelligence*, 7(1):39–55, 1997.

[59] Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*, 2019.

[60] Vivian Lai and Chenhao Tan. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 29–38, 2019.

[61] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Department, Iowa State University, October 1998.

[62] Dongha Lee, Sehun Yu, and Hwanjo Yu. Multi-class data description for out-of-distribution detection. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2020.

[63] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7167–7177. Curran Associates, Inc., 2018.

[64] Antoine Lemay and Sylvain Leblanc. Cognitive biases in cyber decision-making. In *Proceedings of the 13th International Conference on Cyber Warfare and Security*, page 395, 2018.

[65] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[66] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[67] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[68] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

[69] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[70] Thierry A Mara, Stefano Tarantola, and Paola Annoni. Non-parametric methods for global sensitivity analysis of model output with dependent inputs. *Environmental modelling & software*, 72:173–183, 2015.

[71] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859–874, 2010.

[72] Microsoft. Machine learning operations maturity model. `https://docs.microsoft.com/en-us/azure/architecture/example-scenario/mlops/mlops-maturity-model`, 2020. Accessed: 2020-04-12.

[73] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[74] Tim Miller, Piers Howe, and Liz Sonenberg. Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547*, 2017.

[75] Sina Mohseni, Niloofar Zarei, and Eric D Ragan. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *arXiv preprint arXiv:1811.11839*, 2018.

[76] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[77] Douglas C Montgomery. *Design and analysis of experiments*. John wiley & sons, 2017.

[78] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[79] Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*, 2018.

[80] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[81] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pages 3809–3818. PMLR, 2018.

[82] William L Oberkampf, Martin Pilch, and Timothy G Trucano. Predictive capability maturity model for computational modeling and simulation. Technical report, Sandia National Laboratories Albuquerque, NM, 2007.

[83] Art B Owen. Sobol' indices and Shapley value. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):245–251, 2014.

[84] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[85] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, 2016.

[86] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[87] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Towards practical verification of machine learning: The case of computer vision systems. *arXiv preprint arXiv:1712.01785*, 2017.

[88] LeeAnn Perkins, Janet E Miller, Ali Hashemi, and Gary Burns. Designing for human-centered systems: Situational risk as a factor of trust in automation. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 54, pages 2130–2134. SAGE Publications Sage CA: Los Angeles, CA, 2010.

[89] C Petersen and R Lentz. Surfacing critical cyber threats through security intelligence: A reference model for it security practitioners. *SANS Institute*, 2015.

[90] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[91] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K Nicholas. Malware detection by eating a whole exe. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[92] Saman Razavi, Anthony Jakeman, Andrea Saltelli, Clémentine Prieur, Bertrand Iooss, Emanuele Borgonovo, Elmar Plischke, Samuele Lo Piano, Takuya Iwanaga, William Becker, Stefano Tarantola, Joseph H.A. Guillaume, John Jakeman, Hoshin Gupta, Nicola Melillo, Giovanni Rabitti, Vincent Chabridon, Qingyun Duan, Xifu Sun, Stefán Smith, Razi Sheikholeslami, Nasim Hosseini, Masoud Asadzadeh, Arnald Puy, Sergei Kucherenko, and Holger R. Maier. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137:104954, 2021.

[93] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*, 2016.

[94] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[95] Christina Richmond and Pere Lindstrom. Idc security survey: As the job churns. Technical report, 2015.

[96] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[97] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019.

[98] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley, 2008.

[99] Karthik Ramakrishnan Corey Salveson. The ai maturity framework. Technical Report MSU-CSE-06-2, Element AI, 2020.

[100] Julian Sanchez, Wendy A Rogers, Arthur D Fisk, and Ericka Rovira. Understanding reliance on automation: effects of error type, error distribution, age and experience. *Theoretical issues in ergonomics science*, 15(2):134–160, 2014.

[101] Kristin E Schaefer, Jessie YC Chen, James L Szalma, and Peter A Hancock. A meta-analysis of factors influencing the development of trust in automation: Implications for understanding autonomy in future systems. *Human factors*, 58(3):377–400, 2016.

[102] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[103] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[104] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[105] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. In *International Conference on Machine Learning*, pages 9046–9057. PMLR, 2020.

[106] Michael R Smith, Nicholas T Johnson, Joe B Ingram, Armida J Carbajal, Bridget I Haus, Eva Domschot, Ramyaa Ramyaa, Christopher C Lamb, Stephen J Verzi, and W Philip Kegelmeyer. Mind the gap: On bridging the semantic gap between machine learning and malware analysis. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pages 49–60, 2020.

[107] Ralph C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*. Society for Industrial and Applied Mathematics, USA, 2013.

[108] Charles Smutz and Angelos Stavrou. Malicious pdf detection using metadata and structural features. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, page 239–248, New York, NY, USA, 2012. Association for Computing Machinery.

[109] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[110] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[111] Nedim Srndic and Pavel Laskov. Practical Evasion of a Learning-Based Classifier: A Case Study. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pages 197–211, Washington, DC, USA, 2014. IEEE Computer Society.

[112] Mallory C Stites, Megan Nyre-Yu, Blake Moss, Charles Smutz, and Michael R Smith. Sage advice? the impacts of explanations for machine learning models on human decision-making in spam detection. In *International Conference on Human-Computer Interaction*, pages 269–284. Springer, 2021.

[113] David John Stracuzzi, Michael Christopher Darling, Matthew Gregor Peterson, and Maximillian Gene Chen. Quantifying uncertainty to improve decision making in machine learning. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2018.

[114] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[115] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[116] The Institute for Ethical AI & Machine Learning. The responsible machine learning principles. `https://ethical.institute/principles.html`. Accessed: 2020-04-12.

[117] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[118] Laura Toloşi and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994, 2011.

[119] Perry Van Wesel and Alwyn E Goodloe. Challenges in the verification of reinforcement learning algorithms. *National Aeronautics and Space Administration, NASA STI Program*, 2017.

[120] Razvan Viorescu. 2018 reform of EU data protection rules. *European Journal of Law and Public Administration*, pages 27–39, 2017.

[121] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, pages 5133–5142. PMLR, 2018.

[122] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. Evaluating explanation methods for deep learning in security. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 158–174. IEEE, 2020.

[123] Hilde JP Weerts, Werner van Ipenburg, and Mykola Pechenizkiy. A human-grounded evaluation of shap for alert processing. *arXiv preprint arXiv:1907.03324*, 2019.

[124] Eyal Winter et al. The Shapley value. *Handbook of game theory with economic applications*, 3(2):2025–2054, 2002.

[125] Shawn Xu, Subhashini Venugopalan, and Mukund Sundararajan. Attribution in scale and space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9680–9689, 2020.

[126] Weilin Xu, Yanjun Qi, and David Evans. Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers. In *21th Annual Network and Distributed System Security Symposium (NDSS)*, February 2016.

[127] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224, 2004.

[128] Nirit Yuviler-Gavish and Daniel Gopher. Effect of descriptive information and experience on automation reliance. *Human factors*, 53(3):230–244, 2011.

[129] Pourya Habib Zadeh, Reshad Hosseini, and Suvrit Sra. Deep-rbf networks revisited: Robust classification with rejection. *arXiv preprint arXiv:1812.03190*, 2018.

[130] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114, 2004.

[131] H Zhang. The optimality of naive bayes,". In *Proc. Seventeenth Int. Florida Artif. Intell. Res. Soc. Conf. FLAIRS 2004*, volume 1, pages 1–6, 2004.

APPENDIX A. Machine Learning Models

We go over each of the aforementioned models in Chapter 2 and highlight how their mathematical structure might enable them to either identify, or fail to identify, correlations. Experiments are provided to emphasize these ideas.

A.1. Random Forest

Random forests involve a "forest" of decision trees, each producing a "vote" when it comes to classifying a point. To create these trees, the training data is sub-sampled into groups which are then used to train each decision tree [13]. While it is difficult to understand and interpret the ensemble that is the entire forest, individual trees are relatively easy to interpret, especially under the context of how the individual trees are formed.

Given some subset (sampled with replacement) of the training data, each decision tree has to determine a feature and a corresponding threshold for that feature to split on. This determination is made in such a way that the GINI impurity of the resulting splits is minimized.

Definition A.1.1 (GINI Impurity)**.** A measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. A GINI Impurity value of 0 indicates all cases in a node fall into a single class. For J classes with p_i being the proportion of items of class i in the set:

$$I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = 1 - \sum_{i=1}^{J} p_i^2$$

There exist two choices at each split. Which feature to split on, and which value of the chosen feature. Once a feature is known, it is relatively easy to find the value to split on that provides the lowest GINI impurity to each of the resulting sets.

To prevent over-fitting in the forest, the choice of feature for each split is limited to a random subset of all of the possible features (typically of size $\approx \sqrt{|F|}$, where $|F|$ is the number of features). This, for example, stops all of the decision trees from having the same feature split as their top node, if one feature is highly separable. This leads to a diverse set of trees that, hopefully, provide a better approximation of the target function.

To visualize the decision boundaries of individual trees in Figure A-1 a series of decisions over F_1 and F_2 that led to leaf nodes that voted "non-correlated" are plotted. These were pulled from 15/100 trees in a forest where each tree was limited to a depth of 3.

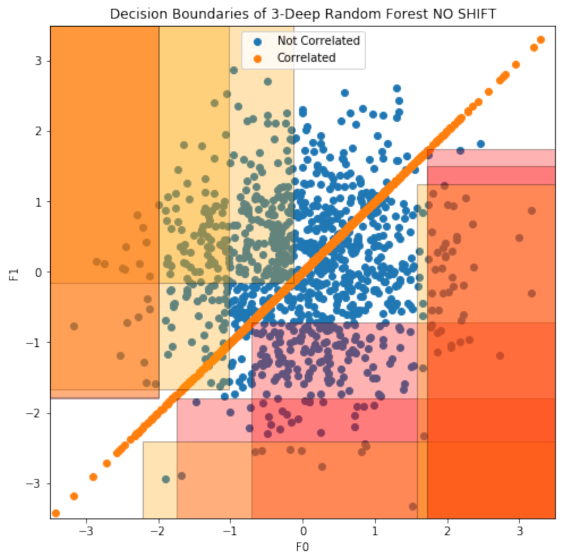


Figure A-1. Example of splits over F_1 and F_2 and the areas that belong to "non-correlated" leaf nodes that result.

Random Forests and Correlations

Now from the results presented Section 2.3.1 it is apparent that random forests are able to "see" correlations as a means of classification. Take the scope of a decision tree with the non-shifted data. Theoretically, no matter the subset of features offered as possible features to split on, there is no "best" feature and no best split. This is because all of the features have *identical* distributions of data (normally distributed with $\mu = 0, \sigma = 1$). Splits that are not at 0 will have different numbers of points between the two resulting groups, but they should have the exact same number of points from each class (identical distributions with respect to each feature individually).

Due to the random nature and variance of sampling, many splits will offer slight decreases to GINI impurity. However this will be completely random, so theoretically each feature should appear $1/|F|$ of the time in the parent node for the non-shifted data (We verify this experimentally in section A.1). However, once either F_1 or F_2 (the correlated features) are split on, the tree has access to the joint distribution of F_1 and F_2. This means that whichever of the two was not split on has the ability to greatly decrease the GINI impurity compared to the remaining features, where the GINI impurity would remain the same for any split, since every joint distribution except for F_1 and F_2 looks identical. Thus the second of the two correlated features will be chosen as soon as it is in the random feature subset to be split on.

Why is a secondary split in a decision tree conditional to the first? This is illustrated in Figure A-2 on the next page. The distributions of F_1 for both classes are given in the upper histogram, where they are from identical distributions. If F_1 is in the random subset of possible features to split on and the best split is at $F_1 = -0.75$ then we have the distribution of the two classes over F_2 with $F_1 \geq -0.75$ in the histogram on the right. This split of points has a distribution that has separability for a group of the non-correlated points, while F_2 as a whole has a distribution similar to that of F_1.

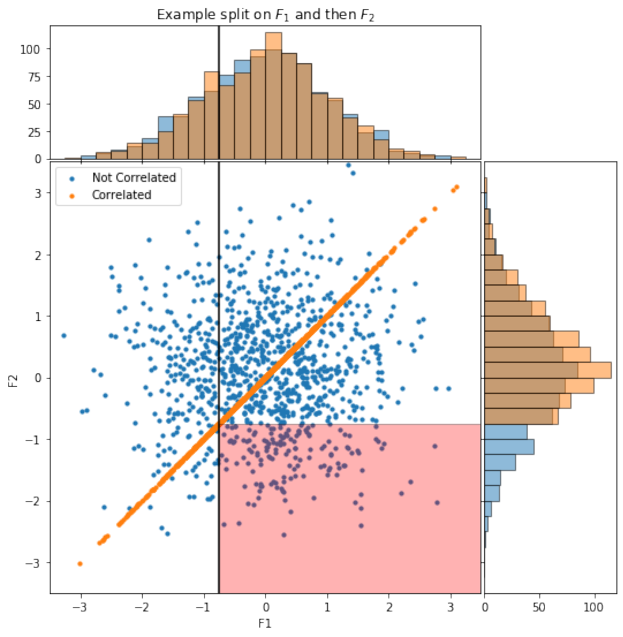


Figure A-2. Example of a split on F_1 (due to random variance) and the resulting distribution with respect to F_2. Note the histogram on the right is for the data to the right of the split at $F_1 = -0.75$.

This difference in distribution comes from the fact that all, or a significant number of (for high correlation values), correlated points with $F_1 \geq -0.75$ will also have $F_2 \geq -0.75$, leaving a "hole" at $F_2 \leq -0.75$. This pattern is seen in practice, as decision trees that contain successive splits on both correlated classes will have the splits taking place on nearly identical thresholds.

Feature Distribution of Parent Node

As mentioned previously, every node in a decision tree has access to a subset of features to make a split on. The default value for this is typically $\sqrt{|F|}$. Since we use 8 features and `int` is a truncation call, every node has access to two of the eight features to determine a split. To verify our earlier claim, that for the non-shifted data, the first node should essentially be random (whether we use 2 features or 8 per node), we repeatedly train random forests and note the feature used in the initial split.

As for the shifted data, when using the default parameters we expect the shifted feature (F_8) to be split on every time it is included in the two features to be selected from. There is a $1/4$ chance that F_8 is in the parent node's two features, where it will be selected every time. Thus we expect F_8 to be used in $1/4$ of the time, with the rest of the features $3/28$ of the time. When all features are used, we expect F_8 to be picked all of the time.

To give a point of comparison, we train the trees using the default number of features to select from at a node, two, as well as with access to all eight features. This is done for each correlation for both the shifted and non-shifted data sets, leading to four sets of results. To eliminate large variances in the results, 100 datasets were generated for each correlation value and random forests for each dataset produced 100 decision trees in their ensemble, leading to a total of 10,000 trees analyzed per correlation value. Table A-1 is for the default parameter for the number of features at a node and Table A-2 on the next page is for nodes using all features.

Table A-1. Counts of instances where a given feature was used in the parent node for a decision tree (out of 10,000), for 8 features per node.

Correlation	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Shifted								
1.0	1088	1131	1074	1064	963	1084	1075	2521
0.9	1032	1058	1108	1089	1027	1093	1070	2523
0.8	1189	1084	1082	1036	1071	1008	1052	2478
0.7	1105	1097	1083	1041	1079	1105	1061	2429
0.6	1051	1069	1087	1156	1017	1027	1105	2488
0.5	1113	1079	1037	1098	1056	1038	1087	2492
0.4	1015	1190	979	1117	1057	1067	1073	2502
0.3	1067	1052	1035	1060	1054	1149	1099	2484
0.2	1119	1051	1134	1044	1015	1086	1034	2517
0.1	1056	1127	1124	1055	1025	1045	1092	2476
0.0	1094	1041	1037	1061	1006	1087	1072	2602
Not-Shifted								
1.0	1285	1265	1220	1267	1231	1250	1226	1256
0.9	1253	1292	1226	1193	1251	1346	1212	1227
0.8	1144	1195	1121	1309	1302	1270	1340	1319
0.7	1213	1263	1253	1217	1310	1267	1269	1208
0.6	1274	1225	1237	1249	1242	1306	1195	1272
0.5	1207	1230	1237	1389	1247	1140	1308	1242
0.4	1280	1256	1270	1255	1234	1261	1240	1204
0.3	1297	1263	1278	1223	1323	1234	1218	1164
0.2	1205	1215	1283	1312	1235	1233	1271	1246
0.1	1251	1252	1249	1210	1297	1262	1341	1138
0.0	1265	1168	1277	1212	1203	1235	1355	1285

From Table A-1 we see our hypothesis was correct. For the shifted data approximately 25% of the time F_8 was chosen in the parent node, while the remaining features were selected $3/28 \approx 10.7\%$ of the time, or 1,070 times of the 10,000 trees. As for the non-shifted data, we see each feature represented approximately 1/8th of the time.

From Table A-2 on the next page we again see our hypothesis was correct. For the shifted data we have that F_8 appears nearly 100% of the time when the node has access to each of the features to make a split. Then for the non-shifted results we again see that the features are represented relatively equally, approximately 1/8th of the time.

It is also important to note that in none of the tests does the correlation value have any effect on the choice of initial feature to split on. This is in-line with the above discussion where each feature looks identical unless a joint distribution is analyzed.

Table A-2. Counts of instances where a given feature was used in the parent node for a decision tree (out of 10,000), for 2 features per node.

Correlation	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Shifted								
1.0	0	0	0	0	0	0	0	10,000
0.9	0	0	0	0	0	0	0	10,000
0.8	0	0	0	0	0	0	0	10,000
0.7	0	0	0	0	0	0	0	10,000
0.6	0	0	0	0	0	0	0	10,000
0.5	0	0	0	0	0	0	0	10,000
0.4	0	0	0	0	0	0	0	10,000
0.3	0	0	0	0	0	0	0	10,000
0.2	0	0	0	0	0	0	0	10,000
0.1	0	0	0	0	0	0	0	10,000
0.0	0	1	0	0	0	0	0	9,999
Not-Shifted								
1.0	1200	1132	1166	1364	1325	1152	1247	1414
0.9	1239	1258	1281	1214	1341	1188	1243	1236
0.8	1322	974	1204	1485	1311	1072	1211	1421
0.7	1289	1118	1432	1416	1140	1502	1160	943
0.6	1320	1375	1223	1356	1261	1243	1068	1154
0.5	1125	1224	1298	1091	1183	1359	1283	1437
0.4	1255	1166	1256	1340	1165	1340	1163	1315
0.3	1378	1129	1133	1222	1206	1375	1169	1388
0.2	1317	1172	1221	1321	1248	1216	1167	1338
0.1	1152	1518	1269	1052	1254	1283	1180	1292
0.0	1246	1203	1341	1292	1038	1367	1376	1137

A.1.1. Multilayer Perceptron

Without getting into the nitty-gritty of the inner workings of a multilayer perceptron we give two different perspectives of the explanation for how an MLP recognizes the presence of a correlation in the data.

Feature-Space Perspective

The most basic reasoning for this phenomenon is understood by how the model partitions the feature space. In Figure A-3 on the following page we show the decision boundary of an MLP trained on the non-shifted data, over F_1 and F_2 with $F_i = 0$ for $i = 3, 4, \ldots, 8$, with $C_{F_1,F_2} = 1.0$.

Here we see a narrow band around $F_1 = F_2$, the width of which expands as the values of these two features increase in magnitude. This expansion simply comes from the lack of training data in that region (typically values lie in the domain $[-3.5, 3.5]$)).

It is also worth noticing the steep gradient that exists near the boundary of the center region. The drop off from 0.99 towards one class to near 0 (for the other class) takes place over a very thin region. This indicates a very sharp confidence in the model, implying that the model just accepts the error for points in the non correlated class that lie near $F_1 = F_2$.

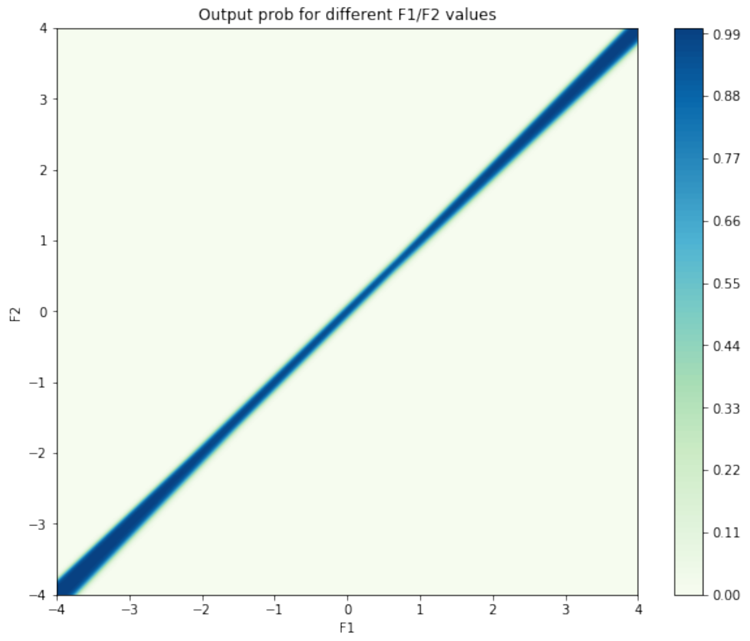


Figure A-3. Output over F_1 and F_2 with F_3 through F_8 set to zero, $C_{F_1,F_2} = 1.0$.

From this perspective we can summarize that the MLP sees a region that contains an extremely high proportion of correlated points. Thus it creates a boundary around that region to identify this class.

Weight Perspective

In an alternate perspective we analyze the weights of the network. Since the MLP consists of a single hidden layer with five hidden nodes the first weight matrix will be 8×5 with 5 bias values. The results of activating on these weights and biases are combined into a single output using an additional 5 weights and a bias. A visualization of these weights and biases is found in Figure A-4 on page 136.

From this we see an interesting pattern in the last three nodes. The weights for F_3 to F_8 in these nodes are near zero, with the weights for F_1 and F_2 alternating in sign between the nodes. We analyze two cases to illustrate how this network sees the correlation, but first we give the weights of the network in Table A-3 on the facing page.

Now, assuming that the random distribution of F_3 through F_8 means that the slightly nonzero weights for these features in the first two nodes are a wash, we can approximate this network in terms of F_1 and F_2, given in Table A-4.

Table A-3. Example MLP weights.

Node	1	2	3	4	5
F_1	-2.3	-1.0	-2.5	3.5	-3.0
F_2	0.7	1.6	2.5	-3.5	3.0
F_3	-0.2	0.0	0.0	0.0	0.0
F_4	0.1	0.0	0.0	0.0	0.0
F_5	-0.4	0.2	0.0	0.0	0.0
F_6	0.3	-0.2	0.0	0.0	0.0
F_7	0.1	0.0	0.0	0.0	0.0
F_8	0.0	0.0	0.0	0.0	0.0
Bias	1.1	1.0	0.0	0.0	0.0
Output	1.1	2.2	-7.5	-8	-8
Bias	-0.6				

Table A-4. Approximation to MLP in terms of F_1 and F_2.

Node	Approximation	$F_1 \approx F_2$
N_1	$-2.3F_1 + 0.7F_2 + 1.1$	$-1.6F + 1.1$
N_2	$-1.0F_1 + 1.6F_2 + 1.0$	$0.6F + 1.1$
N_3	$-2.5F_1 + 2.5F_2 + 0.0$	0
N_4	$3.5F_1 - 3.5F_2 + 0.0$	0
N_5	$-3.0F_1 + 5.0F_2 + 0.0$	0
Output	$A_1 + 2A_2 - 8(A_3 + A_4 + A_5) - 0.6$	where $A_i = \max(0, N_i)$

First, assume $F = F_1 \approx F_2$: From Table A-4 we see that the last three nodes will be zero, as the values of F_1 and F_2 will approximately cancel out. Then if $F \geq 0.7$ we have that N_1 will not activate, but N_2 will. If $F \leq -2$ we have that N_2 will not activate, but N_1 will. If $-2 < F < 0.7$ both will activate. Thus A_1 or A_2 are positive, with $A_3, A_4, A_5 \approx 0$. We can see from the output function that this will be relatively large, as N_1 and N_2 play inverse roles so that at least one of A_1 and A_2 is large. Thus this value will have a sigmoid near 1, matching the correlated class (label of 1).

Second, assume $F_1 \neq F_2$ by a sizable margin and $F = |F_1 - F_2|$: Again from Table A-4 we have that either one or two of the last three nodes will activate, approximately three times the difference between F_1 and F_2. This will then be weighed in the output layer by -8, meaning that the negative contribution to the output is from $-24F$ to $-48F$. Since F has to be decently greater than zero for the point to be non-correlated, it is impossible for the output from the first two nodes to outweigh this (at least under the domain of the training data, as this is seen in the growing region of the correlated class as you get away from $F_1, F_2 = 0$.) Thus the sigmoid of this extremely negative output will be near 0, matching the non-correlated class (label of 0).

From these examples we see that the first two hidden nodes work as a compliment to the last three. The last three nodes have a strongly negative output weight, leading to a non-correlated class if they activate strongly, which happens when $F_1 \neq F_2$. The first two nodes, between the two of them, provide a positive output when $F_1 \approx F_2$. Note that this is *not* an if and only if relationship, it just so happens that when $F_1 \approx F_2$ one or both of these nodes activate. In the case that there is not a correlation but these two nodes activate, the strongly negative contribution from the lack of correlation in the last three nodes will overrule this.

To summarize, the last three nodes recognize the lack of correlation. Then some combination of the first two activate when $F_1 \approx F_2$, for positive or negative values of F_1 and F_2.

Thus in the case of correlation we have low activation from the last three and strong activation from the first two. In the case of no correlation we have a strong, negatively-weighted, activation from the last three that outweighs whatever the activation from the first two is.

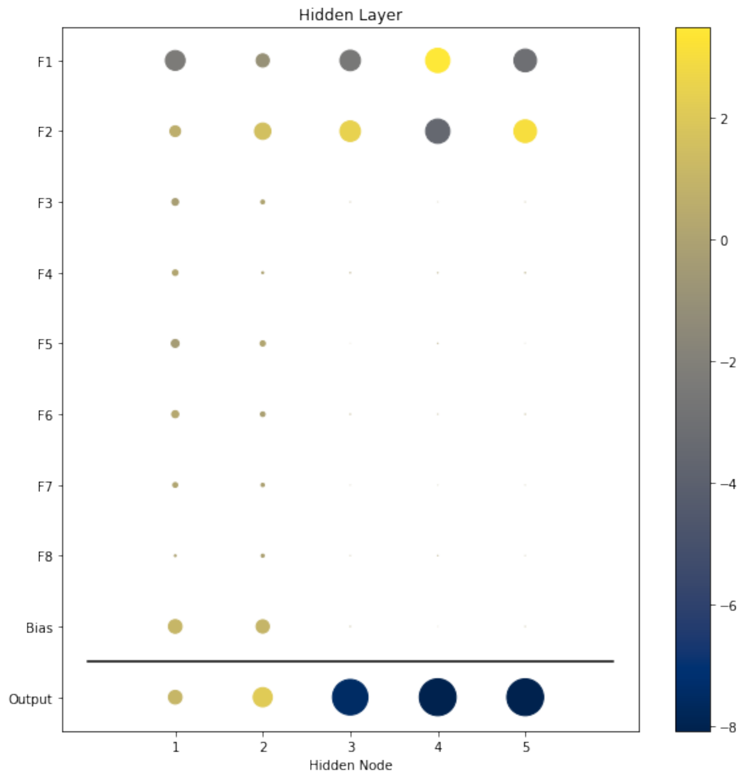
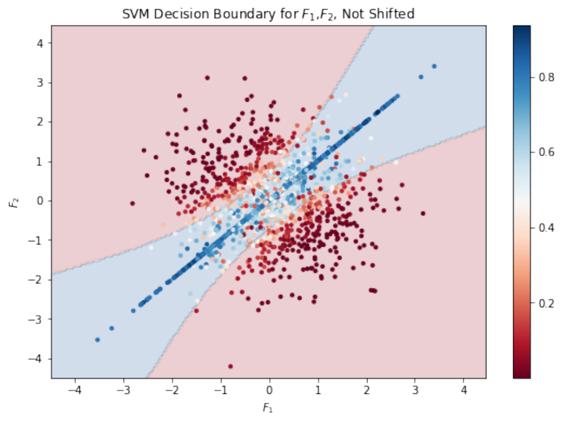
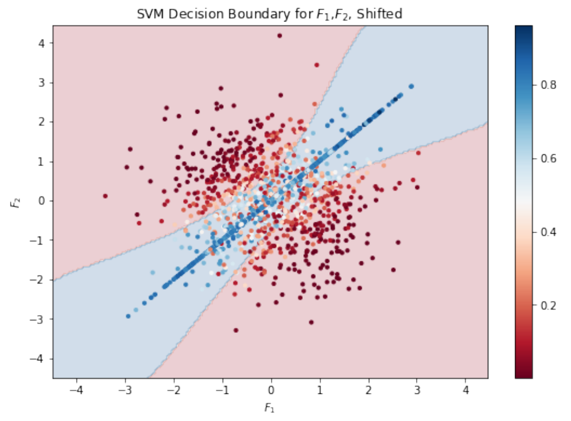


Figure A-4. MLP weights for a single layer with 5 hidden nodes. Size of circle indicates magnitude of the weight, color indicates value.

A.1.2. Support Vector Machine

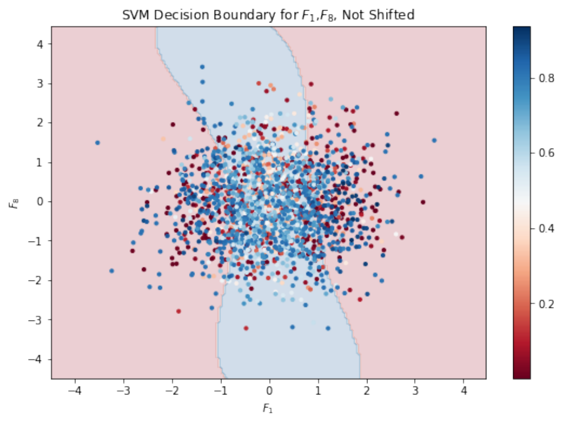
The support vector machine algorithm attempts to create a maximum-separator in the data. This is done by solving a minimization problem with the use a the ’kernel-trick’ to increase separability by projecting the features to a much higher (in the case of the radial basis function kernel, an infinite) dimension. To investigate just how the model understands a correlation we plot the decision boundary along with the prediction probabilities for each point in the training set. In Figure A-5 we give the decision boundary over F_1 and F_2 with the remaining features set to zero, for both the non-shifted and shifted data. In Figure A-6 we give the decision boundary over F_1 and F_8, with the remaining features set to zero, again for both the non-shifted and shifted data.

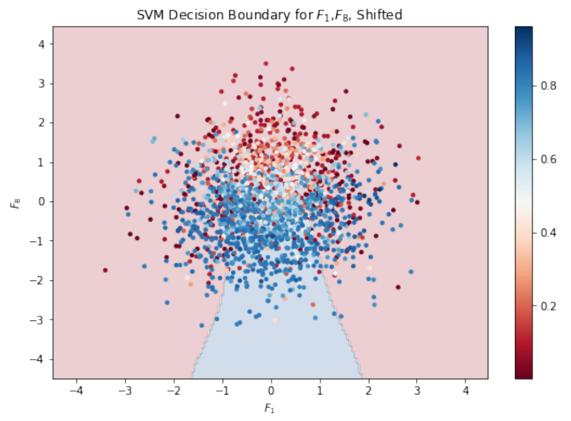




(a) SVM decision boundary over F_1 and F_2, non-shifted. **(b)** SVM decision boundary over F_1 and F_2, shifted.

Figure A-5. SVM decision boundary over F_1 and F_2 for non-shifted (left) and shifted (right) F_8.





(a) SVM decision boundary over F_1 and F_8, non-shifted. **(b)** SVM decision boundary over F_1 and F_8, shifted.

Figure A-6. SVM decision boundary over F_1 and F_8 for non-shifted (left) and shifted (right) F_8.

In these results we see a very similar decision boundary over F_1 and F_2 (Figure A-5). However in the second figure (Figure A-6) we see that the shifted boundary acknowledges that the non-correlated class lies above the correlated class with respect to F_8 and thus provides some means of separability.

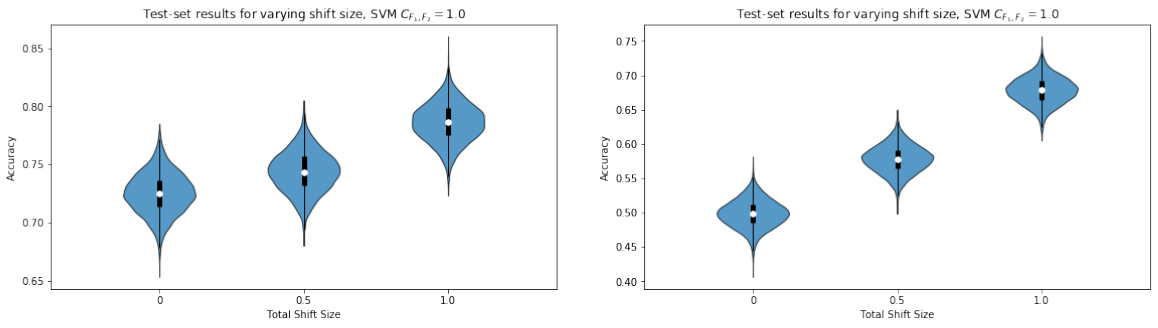
It should be noted that for both sets of plots the color of the scattered points indicate the prediction probability from the model, or how strongly the model believes a point belongs to each class. This comes from a five-fold cross validation during the fit call. Thus these points will integrate information from the higher dimensional space, not just the cross sections of the two displayed.

Varying Shift Size

It was noted that the shifted and non-shifted data had very similar distributions of test-set accuracy scores in the main experiment. To further explore whether this is a failing of the model or just a random occurrence we significantly increase the number of repeats (since SVM is a relatively fast algorithm in our case) and only test the accuracy for SVM over several different shift sizes. Total shifts of 0, 0.5, and 1.0 will be compared.

In Table 2-1 on page 34 it was determined that a best artificial split with respect to F_8 would be at $F_8 = 0$, and it should offer an approximate accuracy of 60%. Following the same idea of an artificial split at $F_8 = 0$, utilizing the CDF of the distribution at 0, ($\mu = -0.5$, $\sigma = 1$) we see that approximately 70% accuracy should be achieved for a pair of means with a separation of 1.

To thoroughly test the effect of larger separation on F_8, a total of 100 data sets for each of the three shift sizes (0,0.5, and 1.0) were generated, and each of those data sets was tested over 100 training/test splits for a total of 10,000 runs per shift size. The test-set accuracy for each shift size in each run was recorded. To generate a baseline the experiment was run for $C_{F_1,F_2} = 1$ and $C_{F_1,F_2} = 0$ to allow comparison to the model's performance when the separability of F_8 is the only real information. The results are displayed in Figure A-7.



(a) SVM Results for various shift sizes, $C_{F_1,F_2} = 1$. **(b)** SVM Results for various shift sizes, $C_{F_1,F_2} = 0$.

Figure A-7. SVM test-set accuracy for shifts on F_8 of total width 0, 0.5, and 1.0. Results are over 10,000 runs. Left: $C_{F_1,F_2} = 1$. Right: $C_{F_1,F_2} = 0$

From these results we see that there is a slight increase in accuracy as the size of the shift increases for the correlated data. However, this increase in accuracy is not directly equal to the increase in theoretical separability on F_8. Looking at the averages over 10,000 runs (Table A-5 on the facing page) we see that there is only a slight increase in accuracy of approximately 4% when the theoretical separation increases 10 fold. We see this theoretical separation in the results for $C_{F_1,F_2} = 0$, as the accuracy increases by exactly 10% as the shift increases from 0.5 to 1.

Table A-5. Mean and standard deviation of SVM test-set accuracies for various shift sizes.

	$C_{F_1,F_2} = 1$		$C_{F_1,F_2} = 0$	
Shift Width	Mean	Std Dev.	Mean	Std Dev.
0.0	0.725	0.01785	0.499	0.02016
0.5	0.744	0.01780	0.578	0.02032
1.0	0.786	0.01682	0.678	0.01956

This muted increase in accuracy when the correlation exists shows that there is some mixing in separability between correlation and feature shift, with respect to a support vector machine. This is evidenced by the marked increase in separability in the results without correlation.

A.1.3. K-Nearest Neighbor

The K-Nearest Neighbor algorithm utilizes the votes (class label) of the closest K points to a given point x to determine the class label of x.

Under this paradigm, the classification of a given point x can be considered with respect to the density of each class in a given area surrounding x. This area, or ε-neighborhood, is variable for each point, as the size of the neighborhood is determined by the distance to the K^{th} neighbor. Thus a point under question, x, is assigned to the class with the highest density in the ε-neighborhood around x.

As seen, and discussed, previously in Figure 2-3 on page 26, the joint distribution of F_1 and F_2 is the only distribution containing information on the separability of the two classes. Visualizing this two-dimensionally under the guise of $C_{F_1,F_2} = 1.0$ we have the correlated class lying on $F_1 = F_2$, with the non-correlated class normally distributed in the surrounding space. We then add in F_3, which has an identical, normal distribution for both classes. This can be seen to distribute the points across the Z-axis, effectively spreading out the points for both classes. However, a hyperplane through $F_1 = F_2$ still holds the entirety of the correlated class, with a handful of class-confused non-correlated points.

Adding in the remaining five dimensions (F_4-F_8) we note that the correlated class still lies on the space defined by $F_1 = F_2$, while the non-correlated class lies on a higher dimension, noisier manifold. It makes sense then that under the Euclidean metric these points become further apart. As an aside, it is noted in [9] that the ratio of the nearest to furthest neighbors to a point approach 1 under modest increases in dimensionality. This curse of dimensionality spreads the points across the higher dimensional space, however the effect of this spreading is that the non-correlated class is spread relatively less, due to $F_1 = F_2$ or $F_1 \approx F_2$ depending on the strength of the correlation.

This difference in separation distance yields the hypothesis that in the space of $F_1 = F_2$, there are seemingly more correlated points in the vicinity than non-correlated points. This leads to an increase in the likelihood that more than half of the K nearest neighbors belong in the correlated class when we consider points in the space $F_1 = F_2$, due to the relative density of this class.

Density Experiments

To explore this idea of higher density of correlated points in the space $F_1 = F_2$ we perform two experiments, following a similar design. First a correlated and non-correlated set of data points are created, 1000 points for each (this is the same as the method used for the main experiments in this paper). Then an additional point is randomly generated from this distribution, x. Given some equally-spaced discretization of the space between -3 and 3 we have a grid of values to step through. Iteratively, F_1 and F_2 for the point x are set to these grid values, and then either the proportion of correlated class points within K neighbors or within some ε-neighborhood are determined. When dealing with the ε-neighborhood case, the total number of points in the neighborhood is also determined. This entire process is repeated to decrease the variance in the results.

To summarize the parameters for these experiments: we control the total number of repeats of the experiment, the granularity of the grid spacing, and the number of nearest neighbors or size of the ε-neighborhood to explore. As well, this experiment can be done for various levels of C_{F_1,F_2} in the data, however in all cases we set $F_1 = F_2 \in \text{GRID}$.

In all of the following results the grid spacing was set to 0.01, the number of repeats was 1000, the number of nearest neighbors used were $K = \{5, 10, 20, 50, 100, 200\}$, and the sizes of the ε-neighborhood were $\varepsilon = \{3, 4, 5, 6, 7\}$.

Results for $C_{F_1,F_2} = 1.0$

First we look at the results for the proportion of correlated class points, Figure A-8 on the facing page. in the K-nearest neighbors we see that with $C_{F_1,F_2} = 1.0$ that more-often-than-not there were more correlated class points within the K neighbors. As we increase the number of neighbors we increase the resolution of the results, however for all cases of $K = 20$ or more we see that near the edges of the distribution that more than half of the neighbors belong to the correlated class. This is due to the nature of the joint distribution. The likelihood of non-correlated point being near an extreme value for $F_1 = F_2$ requires that both F_1 and F_2 are uniquely extreme, essentially squaring the probability of a correlated class point being that extreme in both aspects (as the position of the point with respect to F_1 and F_2 can be considered as coming from a single distribution for $C_{F_1,F_2} = 1.0$).

When we average these 1,000 individual results, Figure A-9 on the next page, we see that *on average* the number of correlated class points in each K-nearest neighbors exceeds the number of non-correlated points. It is interesting to note that the relative proportion of correlated points actually decreases as the number of neighbors examined increases. This illustrates that more points from the normal distribution of the non-correlated class are being included in the neighborhood under the increase of K.

When the ε-neighborhood is considered the results show a similar curve towards a more balanced (with respect to the two classes) center of the grid. It is interesting to note that this balance occurs with a massive increase in the number of neighbors in the ε-neighborhood. This can be seen in Figures A-11 on page 143- A-12 on page 144. With this we see, as before, a higher proportion of

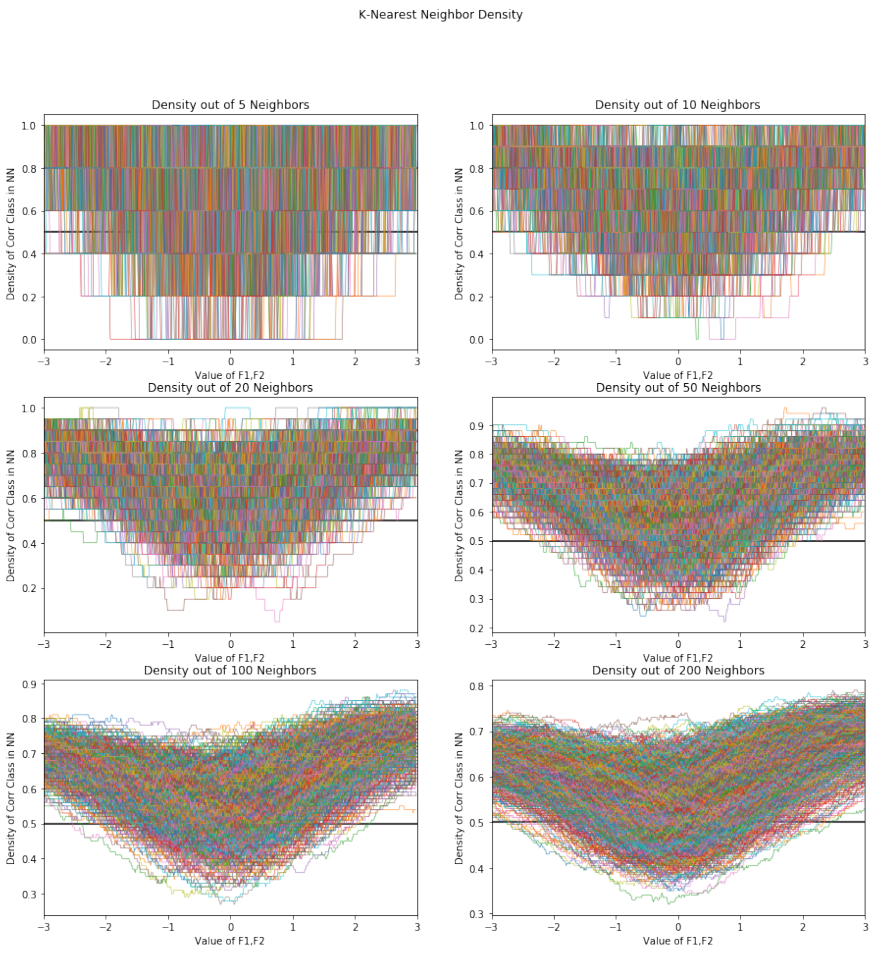


Figure A-8. Proportion of correlated points in K-nearest neighbors at each grid point, for $C_{F_1,F_2} = 1.0$ and $K \in \{5, 10, 20, 50, 100, 200\}$, over 1000 runs.

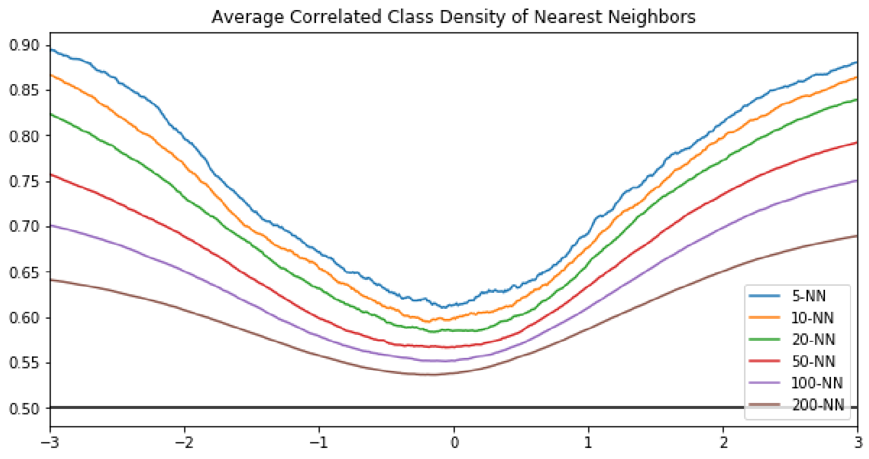


Figure A-9. Average proportion of correlated points in K-nearest neighbors at each grid point, for $C_{F_1,F_2} = 1.0$ and $K \in \{5, 10, 20, 50, 100, 200\}$, over 1000 runs.

correlated points near the ends of the grid, with a ratio of the two classes approaching 0.5 or below in the middle.

As for the averages of the realizations, Figure A-10, we see the correlated class density decrease as the size of the neighborhoods. This is similar to the trend seen under K-nearest neighbors. However, even with this decrease, we see that with neighborhoods of size less than 5.5 (where the total points encompassed by the neighborhood do not max out at 2,000), that the proportion of correlated points remains above 0.5 across the grid.

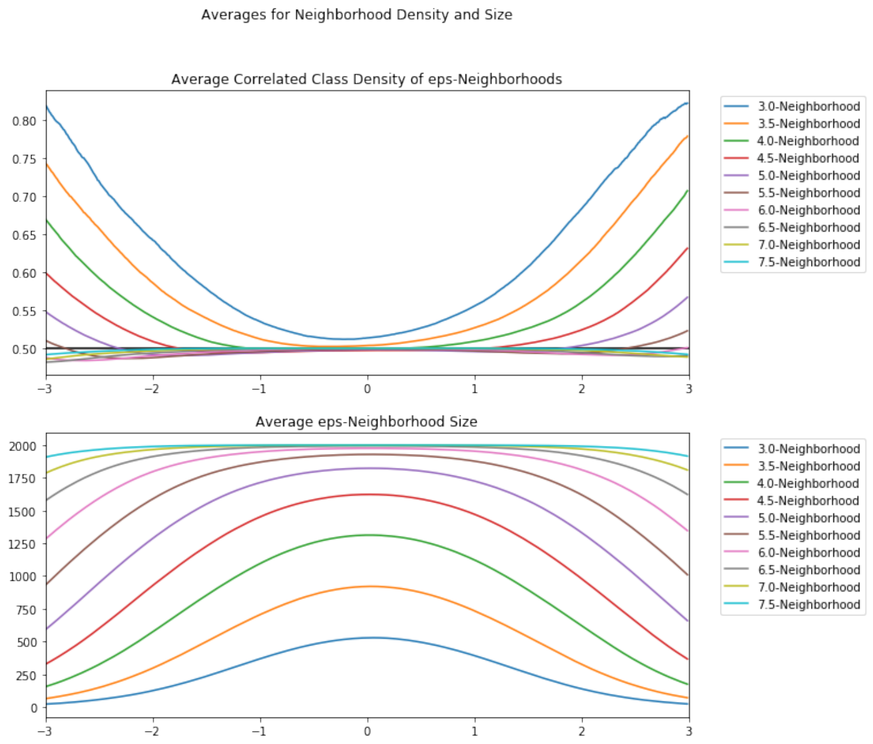


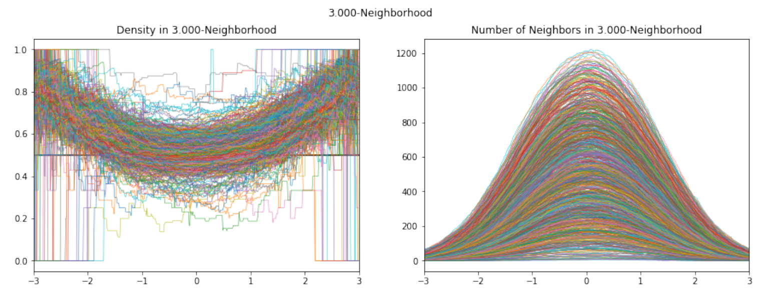
Figure A-10. Average proportion of correlated points in ε-neighbors at each grid point, for $C_{F_1,F_2} = 1.0$ and $\varepsilon \in \{3,4,5,6,7\}$, over 1000 runs.

These results provide further evidence that the density of correlated points in the space surrounding $F_1 = F_2$ lead to the K-nearest neighbor algorithm understanding the correlation in the class, even though the classes overlap in the space.

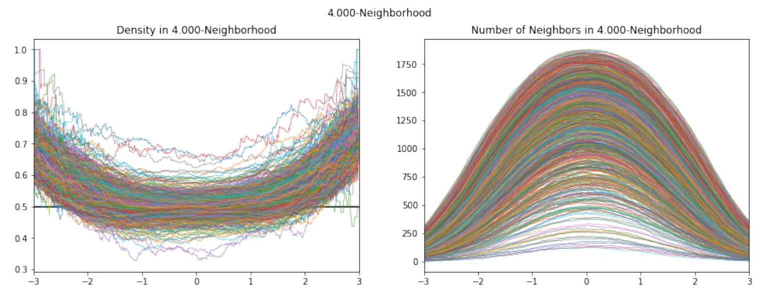
Lesser Correlations

While all of the above was repeated for $C_{F_1,F_2} = 0.9, 0.8, 0.5, 0.2, 0.0$, for the sake of avoiding a second appendix, we leave out the plots except for the average over various K nearest neighbors. These plots can be seen in the notebook `correlated_variable_experiments_knn`. We see the averages for various K neighbors over the different values of C_{F_1,F_2} in Figure A-13 on page 144.

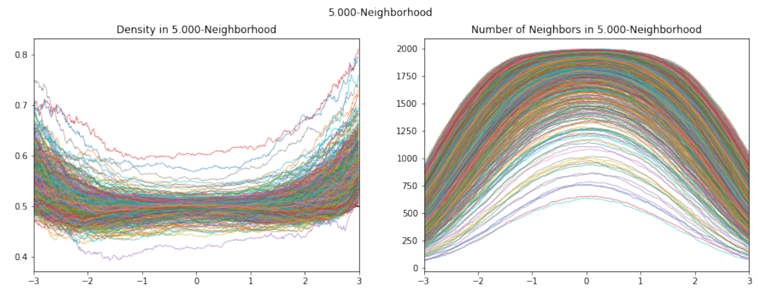
With these results we see a similar trend to $C_{F_1,F_2} = 1.0$ (Figure A-9 on the previous page), where as the number of neighbors increases the average correlated class density decreases. With the



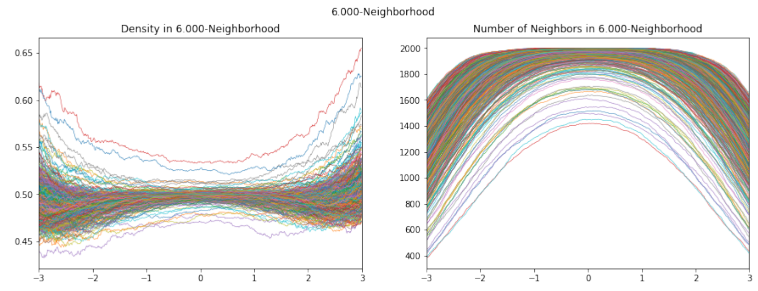
(a) Proportion of correlated points in neighborhood of radius 3.



(b) Proportion of correlated points in neighborhood of radius 4.



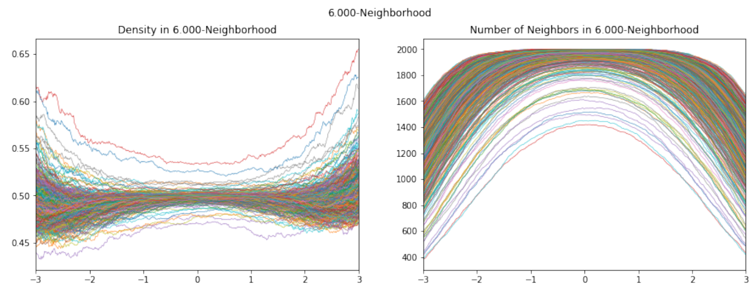
(c) Proportion of correlated points in neighborhood of radius 5.



(d) Proportion of correlated points in neighborhood of radius 6.

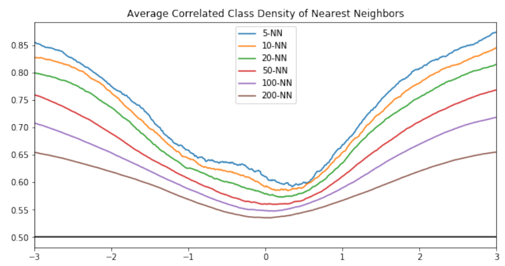
Figure A-11. Proportion of correlated points in ε-neighborhoods for $\varepsilon \in \{3,4,5\}$, realized over 1000 runs.

decrease in correlation we see that the correlated class density for neighbors near points along $F_1 = F_2$ decreases as well. By the time the correlation reaches $C_{F_1,F_2} = 0.5$ we have that the density of the correlated class near $F_1 = F_2$ approaches 0.5. However, this relatively weak correlation still provides an increased density near the edges of the region surveyed. With

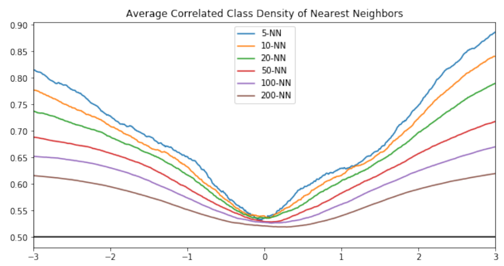


(a) Proportion of correlated points in neighborhood of radius 7.

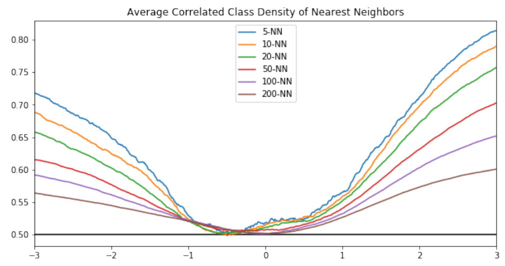
Figure A-12. (Cont.)Proportion of correlated points in ε-neighborhoods for $\varepsilon \in \{3,4,5\}$, realized over 1000 runs.



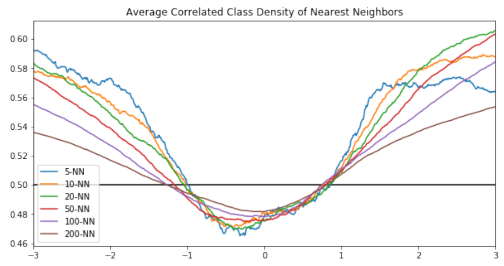
(a) Average proportion of correlated points in K-nearest neighbors, $C_{F_1,F_2} = 0.9$.



(b) Average proportion of correlated points in K-nearest neighbors, $C_{F_1,F_2} = 0.8$.



(c) Average proportion of correlated points in K-nearest neighbors, $C_{F_1,F_2} = 0.5$.



(d) Average proportion of correlated points in K-nearest neighbors, $C_{F_1,F_2} = 0.2$.

Figure A-13. Average proportion of correlated points in K-nearest neighbors at each grid point, for $C_{F_1,F_2} = 0.9, 0.8, 0.5, 0.2$ and $K \in \{5,10,20,50,100,200\}$, over 1000 runs.

$C_{F_1,F_2} = 0.2$ the averages actually drop below 0.5, down to 0.48. This means that many of the points near $F_1 = F_2 = 0$ are essentially evenly distributed for $C_{F_1,F_2} = 0.2$.

When there is no correlation between F_1 and F_2 we see essentially random fluctuations in the density (Figure A-14 on the facing page), as expected. This decrease from high correlated class density to low density in the neighbors provides good evidence for a density based argument for the recognizing of correlations in the data by the K-Nearest Neighbors algorithm.

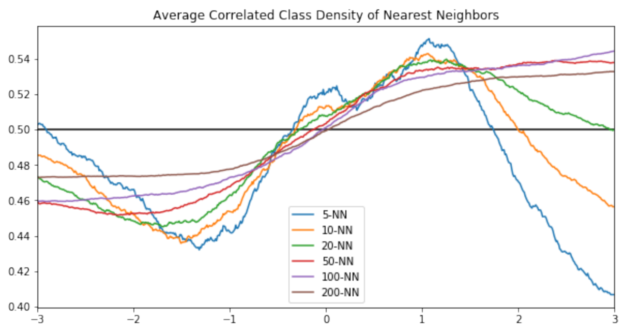


Figure A-14. Average proportion of correlated points in K-nearest neighbors at each grid point, for $C_{F_1,F_2} = 0.0$ and $K \in \{5, 10, 20, 50, 100, 200\}$, over 1000 runs.

A.1.4. Logistic Regression

Logistic regression functions similarly to standard least-squares, in that the algorithm determines weights and a bias such that $t = wx + b$. However, instead of comparing the output t to the target values of y, we push t through a sigmoid function:

$$\sigma(t) = \frac{1}{1+e^{-t}} \tag{A.1}$$

So that it takes on a response in the range $(0,1)$. This can be interpreted as the probability of class membership in a binary class system. Thus we can define the probability that a data point x belongs to class $y = 1$ as

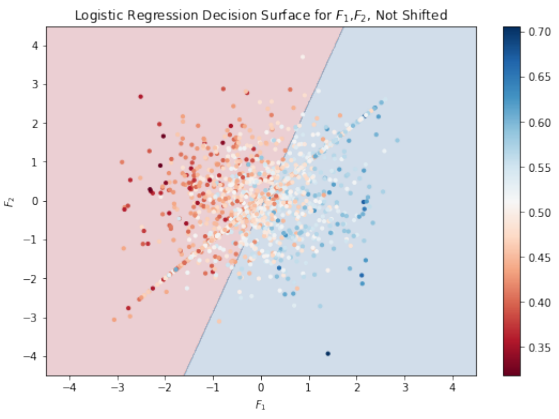
$$P(x) = \frac{1}{1+e^{-(wx+b)}} \tag{A.2}$$

And thus we map every point in the feature space to an output probability for class membership. By the properties of Equation A.2, if $wx + b < 0$ then $P(x) < 1/2$. If $wx + b > 0$ then $P(x) > 1/2$. Since $wx = |w||x|\cos(\theta)$ where θ is the angle between w and x, we have that $\sigma \in (\frac{-\pi}{2}, \frac{\pi}{2})$ will lead to $P(x) > 1/2$. Conversely, $\sigma \in (\frac{\pi}{2}, \frac{3\pi}{2})$ will lead to $P(x) < 1/2$.

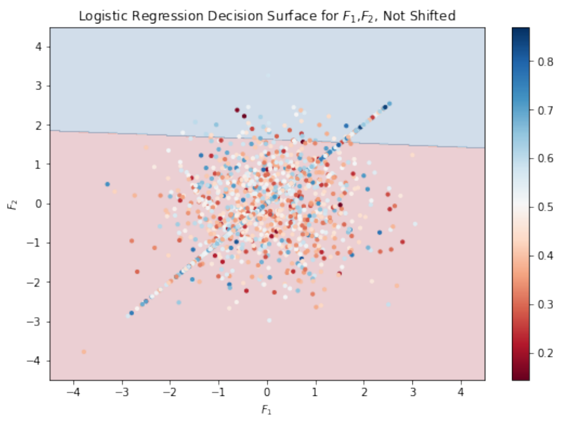
In this context, the vector perpendicular to w will lead to $P(x) = 1/2$, as $\cos(\pi/2) = \cos(3\pi/2) = 0$. Thus everything "above" (with respect to the direction of w) will be more likely to belong to class $y = 1$, with $P(x) > 1/2$ and everything "below" will be more likely to belong to class $y = 0$. Thus we can take this line perpendicular to w as the linear separator of the two classes, with the bias b determining the location of the divider on w.

However, as we understand this data set, a dividing line is going to be unable to capture the region of correlated points, as they lie "inside" the space of the non-correlated class. Thus there is no clear line of separation between the class labels that can be drawn. If we look at the decision boundaries on the surface of F_1 and F_2 (all other components are set to zero) we see this is the case (Figure A-15 on the next page).

In Figure A-15 on the following page on the left we see that in the non-shifted data, for the most part, the individual prediction probabilities (shading of each point) lines up with the decision boundary. Any mismatch comes from the fact that the points are projected down onto



(a) Logistic regression decision boundary over F_1 and F_2, non-shifted.

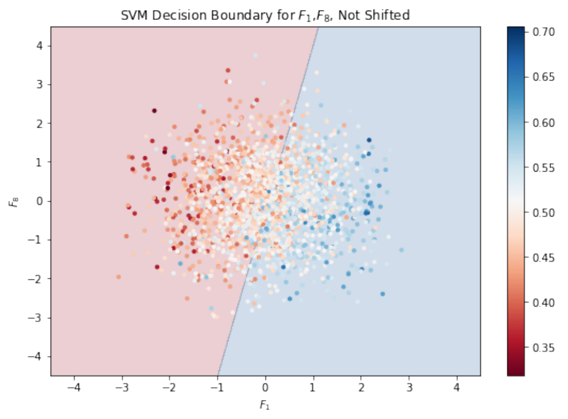


(b) Logistic regression decision boundary over F_1 and F_2, shifted.

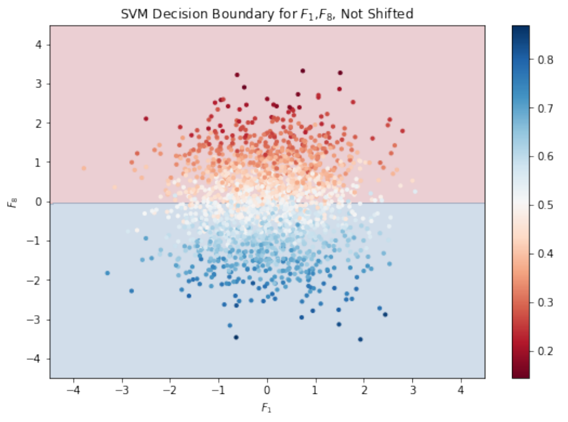
Figure A-15. Logistic regression decision boundary over F_1 and F_2 for non-shifted (left) and shifted (right) F_8, both with $C_{F_1,F_2} = 1.0$. Scatter coloring is sigmoid output for a point.

$F_3, \ldots, F_8 = 0$. We see that the linear separator goes through the correlation line near its center, providing approximately a 50/50 split of the classes. Due to having no meaningful feature separation, with respect to any single feature, the line is essentially a function of the random sampling used to generate the data.

On the right, we see the decision boundary lies above the data, however as seen previously under the analysis of the distributions, it is highly likely that each subdivision of the data consists of an equal distribution of both classes.



(a) Logistic regression decision boundary over F_1 and F_8, non-shifted.



(b) Logistic regression decision boundary over F_1 and F_8, shifted.

Figure A-16. Logistic regression decision boundary over F_1 and F_8 for non-shifted (left) and shifted (right) F_8. Scatter coloring is sigmoid output for a point.

When we analyze the decision surface with respect to F_1 and F_8, with $F_2, \ldots, F_7$ set to zero, we see a similar result for the non-shifted data. The actual labels are distributed evenly between the data, resulting in a division that achieves approximately 50% accuracy. When we look at the shifted data on the right, we see a much clearer picture of separability. It should also be noted that a large magnitude weight on F_8 means that higher and lower probabilities for class membership are achieved as points get away from the line near $F_8 = 0$. Although there is only a slight increase in accuracy (up to 60% from 50%), the shifted data provides a means for the linear separator to apply a meaningful distinction between the classes, rather than being decided by randomness as in the non-shifted data.

Feature Importance

For the non-shifted data an example of w and b are:

$$w = \begin{bmatrix} 0.157 & -0.058 & -0.064 & 0.100 & -0.070 & 0.092 & -0.024 & -0.037 \end{bmatrix} \qquad b = -0.011 \tag{A.3}$$

Whereas for the shifted data:

$$w = \begin{bmatrix} 0.001 & 0.016 & -0.075 & -0.128 & -0.086 & -0.007 & -0.008 & -0.506 \end{bmatrix} \qquad b = -0.027 \tag{A.4}$$

In the non-shifted data we see the largest magnitude feature weight belongs to F_1, however it is of similar magnitude to F_4 and F_6. In the case of the shifted data, F_8 is the largest magnitude feature, and approximately four times larger than the next largest in magnitude, F_4, as well as approximately four times larger than the largest feature in the non-shifted data.

This indicates that F_8 is the most important feature in the shifted data, which aligns with the increased separability coming from F_8.

A.1.5. Gaussian Naive Bayes

Naive Bayes revolves around maximizing the probability of a class label y given some input data $x_1, \ldots, x_n$ where n is the number of features $|F|$. Using Bayes' Theorem we have:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_2)} \tag{A.5}$$

Now, making the assumption that all features are independent, we have that the probability of x_i taking a value given y and all other features is the same as the probability as x_i taking that value given y. Thus:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y) \tag{A.6}$$

Using this assumption of independence we can replace $P(x_1, \ldots, x_n|y)$ with $\prod_{i=1}^{n} P(x_i|y)$ in the numerator of Equation A.5. Noting that the denominator of this equation is a constant with respect to the input data we can remove the denominator and replace the equality with proportionality. Thus we find the class that maximizes the probability:

$$P(y|x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{A.7}$$

Which gives the actual problem:

$$\hat{y} = \text{argmax}_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{A.8}$$

As for the Gaussian part of Gaussian Naive Bayes, we assume a Gaussian distribution of each feature with respect to the class label and thus derive

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{A.9}$$

This derivation comes from the work in [131].

Essentially, given some probability of a value occurring in a feature for each y, we take the y that is most likely over all features (product) and most likely without respect to any x_i ($P(y)$).

Why does this stop Naive Bayes from learning correlation? The actual maximization problem revolves around $P(y)$, which is simply the relative frequency of each class (which are equal, both classes have the same frequency), and the product of all $P(x_i|y)$. The "winning" class label will have higher probabilities for some or all of the features.

However, our data is drawn from a normal distribution with $\mu_i = 0$ and $\sigma_i = 1$ for $i = 1, 2, \ldots, 8$ *for both classes*. So $\mu_{i,y} = 0$ for all i and both y (with the same holding for $\sigma_{i,y}$). Thus the values for $P(x_i|y)$ given in Equation A.9 will all approach

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_{i,y}^2}} \exp\left(-\frac{(x_i - \mu_{i,y})^2}{2\sigma_{i,y}^2}\right) \tag{A.10}$$

$$P(x_i|y) = P(x_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_i^2}{2}\right) \tag{A.11}$$

$$\tag{A.12}$$

as the number of samples increases so that the observed values for $\mu_{i,y} \to 0$ and $\sigma_{i,y} \to 1$ for all i and y. Thus $P(x_i|y)$ are all just functions of x_i, no matter the specific feature i or class label y.

This means that the actual class allocation comes from variance in the sampling of the normal distribution, meaning that some means differ slightly from zero and some variances differ slightly from one. This means the labels are essentially a function of random sampling, and thus we can expect the accuracy to be around 0.5, as seen in the results.

Shifted Data

For the shifted data, in F_8 we actually see a difference in values of $P(x_8|y)$. Since the correlated class ($y = 1$) has $\mu_{8,1} = -0.25$ and the non-correlated class ($y = 0$) has $\mu_{8,0} = 0.25$ they will have different conditional probabilities (note that they both have $\sigma_{8,0} = \sigma_{8,1} = 1$):

$$P(x_8|y = 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i + 0.25)^2}{2}\right) \tag{A.13}$$

$$P(x_8|y = 0) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i - 0.25)^2}{2}\right) \tag{A.14}$$

In Figure A-17 on the next page we see that the conditional probability for $P(x_8|y = 1)$ is higher when $x_8 < 0$ and $P(x_8|y = 0)$ is higher when $x_8 > 0$. Since all $P(x_i|y = 0) \approx P(x_i|y = 1)$ for $i = 1, 2, \ldots, 7$ and $P(y = 0) \approx P(y = 1)$, we have that the label for a class will be determined by whether x_8 is greater than or less than 0.

Our results then align with the discussion surrounding Table 2-1 on page 34, where it was determined that F_8 has approximately 60% accuracy in separability when $F_8 = 0$ is used as a separator, as it will be in Gaussian Naive Bayes for the shifted data.

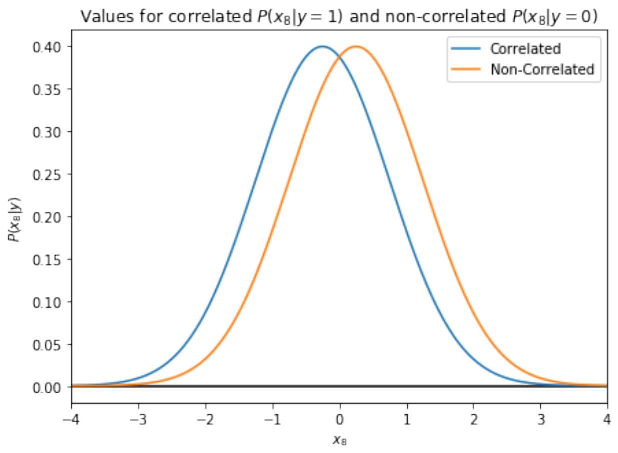


Figure A-17. Values for $P(x_8|y=1)$ and $P(x_8|y=2)$ in the shifted data, where $y=1$ in the correlated class ($\mu_{8,1}=-0.25$) and $y=0$ in the non-correlated class ($\mu_{8,0}=0.25$)

APPENDIX B. Closed-Form Analytical Solution to Shapley Values

In this Section, we briefly provide the details for the closed-form analytical solution to the Shapley values that are used for the fidelity calculations. Calculating Shapley values following Equation 4.3 is fairly straight forward once a value function is established. Here, we provide the value function that we used.

Following the proof provided by Aas et al. [1] for value function of a linear equation $f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j \mathbf{x}_j$ where x_j^* represents the an observed value of x_j for a data point $\mathbf{x}$:

$$\begin{aligned} v|_{\mathbb{S}} &= \mathbb{E}\left[f|_{\mathbb{S}}\right] \\ &= \sum_{j \notin \mathbb{S}} \beta_j \mathbb{E}[x_j] + \sum_{j \in \mathbb{S}} \beta_j x_j^*. \end{aligned} \tag{B.1}$$

Here, we use variance as our uncertainty measure for the QoI and for the input features in determining the global importance values. Therefore, we use the standard deviation σ for the x_j in $\mathbb{S}$ using the nominal values defined in Equation 4.9 in place of actual values for x_j^*. We calculate the expectation $\mathbb{E}[x_j]$ using the mean values μ for the variables not in $\mathbb{S}$. As all of the features have a zero mean, $\mathbb{E}[x_j] = 0$ meaning that the first summation in Equation B.1 is zero and $Var(x_j) = \mathbb{E}[x_j^2]$.

B.0.1. Value of $v|_{\mathbb{S}}$ for Linear Models and Independent Variables

For the the linear model (Equation 4.7), the value function for each subset $\mathbb{S}$ in $\mathbb{M}$ is:

$$\begin{aligned} v|_{\emptyset} &= 0 \\ v|_{x_j} &= \left(\beta_j \sigma_j\right)^2 \\ v|_{x_j, x_k} &= \left(\beta_j \sigma_j\right)^2 + \left(\beta_k \sigma_k\right)^2 \\ v|_{x_j, x_k, x_m} &= \left(\beta_j \sigma_j\right)^2 + \left(\beta_k \sigma_k\right)^2 + \left(\beta_m \sigma_m\right)^2 \\ v|_{x_1, x_2, x_3, x_4} &= \left(\beta_1 \sigma_1\right)^2 + \left(\beta_2 \sigma_2\right)^2 + \left(\beta_3 \sigma_3\right)^2 + \left(\beta_4 \sigma_4\right)^2 \end{aligned} \tag{B.2}$$

B.0.2. Value of $v|_{\mathbb{S}}$ for Linear Models and Correlated Variables

When there are correlated features, knowing the value of one feature provides information for the expected value for the features that are correlated with it. Thus, if feature x_k is perfectly correlated with x_j the mean and variance x_k are equivalent to those of x_j. If x_1 and x_2 are perfectly correlated, the value functions are the same as above in Equation B.2 except for the following cases when either x_1 or x_2 are in $\mathbb{S}$ since one feature provides information about the other correlated feature when it is not included in $\mathbb{S}$:

$$\begin{aligned}
v|_{x_1} = v|_{x_2} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 \\
v|_{x_1,x_3} = v|_{x_2s,x_3} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 + (\beta_3\sigma_3)^2 \\
v|_{x_1,x_4} = v|_{x_2,x_4} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 + (\beta_4\sigma_4)^2 \\
v|_{x_1,x_3,x_4} = v|_{x_2,x_3,x_4} = v|_{x_1,x_2,x_3,x_4} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 + (\beta_3\sigma_3)^2 + (\beta_4\sigma_4)^2
\end{aligned} \tag{B.3}$$

B.0.3. Value of $v|_{\mathbb{S}}$ for Nonlinear Models and Independent Variables

For the nonlinear model (Equation 4.8), the value function changes for x_3 and x_4 from Equation B.2 due to their multiplicative relationship and an expected value of $\mathbb{E}[x_j] = 0$. Therefore, the series of equations changes from Equation B.2 accordingly:

$$\begin{aligned}
v|_{x_3} = v|_{x_4} &= 0 \\
v|_{x_1,x_3} = v|_{x_1,x_4} &= (\beta_1\sigma_1)^2 \\
v|_{x_2,x_3} = v|_{x_2,x_4} &= (\beta_2\sigma_2)^2 \\
v|_{x_3,x_4} &= (\beta_3\sigma_3)^2(\beta_4\sigma_4)^2 \\
v|_{x_1,x_2,x_3} = v|_{x_1,x_2,x_4} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 \\
v|_{x_1,x_3,x_4} &= (\beta_1\sigma_1)^2 + (\beta_3\sigma_3)^2(\beta_4\sigma_4)^2 \\
v|_{x_2,x_3,x_4} &= (\beta_2\sigma_2)^2 + (\beta_3\sigma_3)^2(\beta_4\sigma_4)^2 \\
v|_{x_1,x_2,x_3,x_4} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 + (\beta_3\sigma_3)^2(\beta_4\sigma_4)^2
\end{aligned} \tag{B.4}$$

B.0.4. Value of $v|_{\mathbb{S}}$ for Nonlinear Models and Correlated Variables

Finally, for a nonlinear model where x_1 and x_2 are correlated, we get a combination of Equations B.3 and B.4 producing:

$$\begin{aligned}
v|_{x_1} = v|_{x_2} = v|_{x_1,x_2} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 \\
v|_{x_3} = v|_{x_4} &= 0 \\
v|_{x_1,x_3} = v|_{x_1,x_4} = v|_{x_2,x_3} = v|_{x_2,x_4} = v|_{x_1,x_2,x_3} = v|_{x_1,x_2,x_4} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 \\
v|_{x_3,x_4} &= (\beta_3\sigma_3)^2(\beta_4\sigma_4)^2 \\
v|_{x_1,x_3,x_4} = v|_{x_2,x_3,x_4} = v|_{x_1,x_2,x_3,x_4} &= (\beta_1\sigma_1)^2 + (\beta_2\sigma_2)^2 + (\beta_3\sigma_3)^2(\beta_4\sigma_4)^2
\end{aligned}$$

APPENDIX C. Details for Classification Trust

We provide additional details for clarity.

C.0.1. Scaling and Measuring Distance

Our measures can be used with any scaling, but scaling is important and unavoidable. Doing no scaling is a decision to use the scaling inherent in the raw data. Our algorithms rely on the distance between a pair of points, which combines coordinate value differences into a single number. Each coordinate of a point corresponds to a different feature. For some data, such as the intermediary layers in a deep neural network, the features may already be comparable. For other data, such as tabular text, each might be expressed in different units, and at different scales. This can dramatically bias the value of a distance, with coordinates at large scales dominating. In the absence of domain-specific or problem-specific information, we deem it reasonable to weight the importance of each feature equally. Our approach is to take the min and max value of each feature in the training data, and linearly scale it to $[0, 1]$. Thus, each feature is scaled to a fraction of its range that appears in the training data. Query points must be scaled the same way for consistency, but there are no limits on their values and their coordinates may fall outside $[0, 1]$.

Categorical Features

Categorical features have values from a non-ordinal set: e.g., red, yellow, blue. We map these to mutually exclusive binary variables: e.g., is-red, is-yellow, is-blue. Of course, for any point, exactly one of these binary variables will be True.

Binary Features

False and True are mapped to 0 and 1.

C.0.2. Is This Just Another Classifier?

We introduced geometric measures of how trustworthy a classification is. We believe it is a benefit that these measures are classifier-agnostic, and can serve as a check on any supposed confidence that a classifier self-reports. We believe it is a benefit that these measures are classifier-generic, and can be used to compare the trustworthiness of different types of classifiers, including those not yet invented.

Any resemblance to an existing classifier is purely coincidental.

APPENDIX D. Ensemble of Experts Baseline

D.1. Ensemble of Experts Baseline

We sought to develop an expert baseline explanation for comparison to other methods by combining the decision rationale of multiple domain experts. This study was conducted using the PDFrate classifier [108], specifically using the publicly published dataset [1] which uses the mimicus feature extractor [111]. The PDFrate classifier uses metadata and structural features to identify malware embedded in PDF files using a Random Forest model. Five experts were included in this study representing a wide range of experience (from 2 to 10 years) and familiarity with PDF malware.

Experts were provided with metadata from 20 PDF samples and asked to provide a classification (benign, malicious, or don't know), confidence (low, medium, or high), and two lists of attributes that indicate each class. The metadata provided is the exact same metadata from which the PDFrate features are extracted. However, analysts were asked to provide classification based on their own intuition so the analysts may have focused on different attributes than the PDFrate classifier.

We normalized the identified attributes from each analyst, combining equivalent properties into a single feature. For example, the following shows an example of one sample from an inexperienced analyst:

```
Sample Hash:
 1a2e2f3a722fd4614f60ba35696e5042bbbef72415fd5dca10cf56d5af11bbcb
Classification: malicious
Confidence: medium
Benign Attributes: Large size
Malicious Attributes: Possible suspicious timezone, javacript
 presence, acroform, URL
```

The following is the classification and explanation from an experienced analyst for the same PDF:

```
Sample Hash:
 1a2e2f3a722fd4614f60ba35696e5042bbbef72415fd5dca10cf56d5af11bbcb
Classificaton: malicious
```

[1] https://github.com/csmutz/pdfrate

```
Confidence: high
Benign Attributes:
  NA
Malicious Attributes: Javascript, Acroform, +800 TZ, Limited
  content, Single page, Small number of images, text, etc., file
  and link to file (and inidcator of hacking tools masm), Lack
  of metadata (creator, author, etc), Mismatches in metadata:
  Ex. different date formats, Repeated fields but different
  values for uuids
```

An example of the consolidated features on the same PDF from the five analysts is as follows, with features reported by multiple analysts indicated by number in parenthesis:

```
Classification
  malicious (5)
Confidence
  medium (4)
  high
Benign Attributes
  Large file size
  Medium file size
  Large amount of content
Malicious Attributes
  +800 timezone (5)
  Presence of Javascript (5)
  Presence of Acroform (3)
  Small amount of content
  single Page
  Images
  Text
  Presence of URL (5)
  URL target is local file
  masm tools (3)
  Lack of Metadata (2)
  mismatch metadata
  Negative date delta
  Presence of small boxes
  Presence of embedded document objects
```

Some qualitative observations from compiling this data is:

- Analysts classifications were very consistent
- Analysts frequently identified the same attributes, but occasionally listed them as support for opposing classes

Count	Analyst Feature Coverage	Number of ML Features Required
1	complete	many
8	complete	few
33	complete	one
2	poor	many
37	poor	few
1	poor	one

Table D-1. Counts of analyst features by how well they are represented by the machine learned feature set and how many of the machine leared features are required to provide coverage

- Analyst experience level wasn't strongly correlated to higher accuracy or confidence
- Analyst experience was correlated with longer explanations (more attributes)

The attributes identified by analysts do not align perfectly with the features used by the machine learning model in all cases. Some are nearly identical. For example, the "Presence of Javascript, which is the top analyst feature corresponds directly with count_js and count_javascript, the 2nd and 3rd ranked features of the machine learned model. However, in many cases the human attributes are higher level or more abstract concepts while the features for the machine learner are necessarily easily quantifiable. For example, one commonly identified analyst feature is "Lack of Metadata". This concept is well represented in machine learner features as the count of characters in each metadata field, but this requires utilizing multiple machine learner features. Other abstract and complex analyst features, such as "Normal looking metadata" are not fully represented by features in the machine learning model, even though portions of these concepts are reflected in features such as histogram analysis of text in metadata fields. Table D-1 was created based on input from the primary PDFrate developer indicating how well the analyst features are represented in the machine learning feature set and how many machine learning features are required to replicate the analyst feature.

D.2. Ensemble of Experts Comparison

We compared the baseline of expert ensemble explanations to SHAP_Dep and SHAP by measuring the intersection of the top 5 features per prediction class as identified by each method. For the expert ensemble, features were ranked by the number of analysts that listed the feature for a given class. For SHAP_Dep, the full feature space was evaluated for correlations. For SHAP, kernelshap with the full training set provided as sample data was utilized. We limited the number of features included by each method to a threshold that ensured that each method had approximately the same number of most important features in each explanation.

The similarity in explanations was evaluated by having the same domain expert that compiled the expert ensemble data compare the features in each explanation. If the expert explanation had a feature that was fully or mostly indicated by a feature in the machine learning feature set, then the two explanations were considered to intersect on that feature pair. For example, if the expert explanation included "Presence of Javascript", then it would intersect with other explanations methods if either or both of the features count_js or count_javascript were present in the list of important machine learning model features. Note that this means that only expert features that were completely represented by one or few machine learning features could possibly overlap. 70% of the occurances of features present in the expert explanations were able to be represented by ML features, indicating that the majority of the features identified by experts could have intersected with the machine learning model features.

The expert explanations overlapped with 20% of the features in SHAP explanations compared to 8% for SHAP_Dep. This difference likely indicates that SHAP is more effective than SHAP_Dep in creating explanations that match human analyst intuition. The expert that compiled these results also indicated that the features identified by the SHAP_Dep seemed consistent with a greater emphasis on correlated features. The domain expert noticed that some of the features identified by experts were related to groups of features related to each other. For example, in one sample, the expert ensemble identified that a specific metadata value was important. This specific value can not be directly represented by numerical features so it is not directly represented in the machine learning model feature set. There are various model features that are derived from the metadata field including the counts of various types of characters (lower case, upper case, special, etc). The SHAP_Dep explanation included many of these features derived from the same metadata as the field identified by the expert ensemble. In the cases where specific values or terms are important for classification but parameterized features are not specific enough to capture individual values, it is expected that multiple related features could capture these terms and those features would necessarily be correlated. That SHAP_Dep identified presumably correlated features related to specific terms identified by the expert ensemble indicates that including correlations in explanations may be useful in some situations.

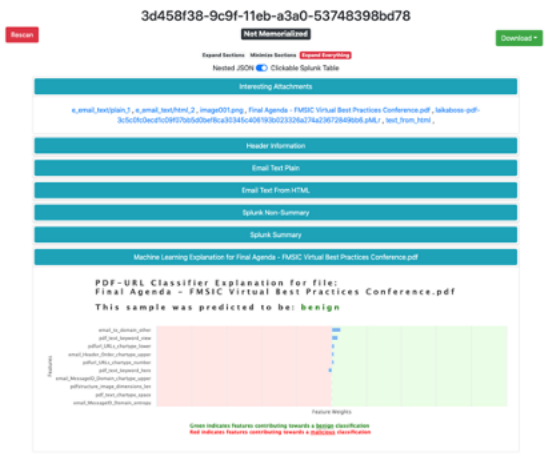
Both SHAP and SHAP_Dep had very low overlap with expert explanations (20% and 8% respectively). It is recognized that the experts may focus on different aspect of the samples than the machine learning model, but it was possible for the machine learning model to represent 70% of the features in the expert explanations. The lack of consistency between machine learning model and expert explanations seems to indicate that model explanations may not be intuitive for analysts. This apparent semantic gap between analyst intuition and model rationale indicates that model explanations may not be useful for aiding analyst comprehension of observations whether it be due to deficiencies in explanation methods or differences in model and expert rationale. Furthermore, SHAP and SHAP_Dep explanations were very different. SHAP and SHAP_Dep explanations overlapped less than 20%.

Given these results, future work should address these differences when providing explanations. Previous research in social science has found that explanations that do not match human intuition are often disregarded [73]. Thus, some training or further explanations of unexpected explanations from the ML model may be required for end-user adoption. This also helps motivate the difference between using human-based explanation metrics, which may be biased to the way humans understand how a model works versus what a ML is actually doing.

APPENDIX E. Interview Questions

*** The following questions may be asked in a Delphi Method, but likely in a phone interview ***

1. Please describe your job role(s) as it pertains to model maintenance.
2. What is your primary goal when performing model maintenance? How do you know when you've achieved it?
3. How do you normally interact with the pdf-url & macro machine-learning models / outputs? About how often?
4. What other machine learning models do you interact with in this way?
5. What information would you normally be searching for when you're investigating the ML output (e.g. classifier = malicious/benign)? What questions do you normally ask in your head as you're doing this? [SHOW EXPLANATION – Figure E-1]
6. When you see this presentation, what do you think it means?
   a) What do you think the features represent? The feature values?
   b) What do you think the direction of the bar represents?
   c) What do you think the color represents?
7. Given this visual, can you imagine any difficulties in obtaining the information you need?
8. Is the amount of information presented appropriate for your needs in evaluating the model output as a model maintainer? If not, please describe.



3d458f38-9c9f-11eb-a3a0-53748398bd78

e_email_text/plain_1 , e_email_text/html_2 , image001.png , Final Agenda - FMSIC Virtual Best Practices Conference.pdf , laikaboss-pdf-3c5c0fc0ecd1c09f37bb5d0bef8ca30345c406193b023326a274a23672849bb6.pMLr , text_from_html ,

Header Information
Email Text Plain
Email Text From HTML
Splunk Non-Summary
Splunk Summary
Machine Learning Explanation for Final Agenda - FMSIC Virtual Best Practices Conference.pdf

PDF-URL Classifier Explanation for file:
Final Agenda - FMSIC Virtual Best Practices Conference.pdf

This sample was predicted to be: benign

email_to_domain_other
pdf_text_keyword_view
pdfurl_URLs_chartype_lower
email_Header_Order_chartype_upper
pdfurl_URLs_chartype_number
pdf_text_keyword_here
email_MessageID_Domain_chartype_upper
pdfstructure_image_dimensions_len
pdf_text_chartype_space
email_MessageID_Domain_entropy
Features
Feature Weights

Green indicates features contributing towards a benign classification
Red indicates features contributing towards a malicious classification

Figure E-1. Interface of the explanation integrated into the standard user interface used the incident responders.

DISTRIBUTION

Email—Internal [redacted]

Name	Org.	Sandia Email Address
Courtney S. Dornburg	5954	ccdornb@sandia.gov
Curtis Johnson	5952	cjohnso@sandia.gov
Tu-Thach Quach	9364	tong@sandia.gov
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.