GENERALIZED ROBUST FEATURE SELECTION

THESIS

Bradford Lott, First Lieutenant, USAF

AFIT-ENS-MS-22-M-148

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-22-M-148

GENERALIZED ROBUST FEATURE SELECTION

THESIS

Presented to the Faculty

Department of Operations Research

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Bradford Lott, B.S. Biomathematics

First Lieutenant, USAF

March 24, 2022

AFIT-ENS-MS-22-M-148

GENERALIZED ROBUST FEATURE SELECTION

THESIS

Bradford Lott, B.S. Biomathematics
First Lieutenant, USAF

Committee Membership:

Mark A. Gallagher, Ph.D
Chair

Bruce A. Cox, Ph.D
Member

Genearlized Robust Feature Selection

Bradford L. Lott[a], Mark A. Gallagher[a], Bruce A. Cox[a]

[a]*Air Force Institute of Technology Department of Operational Sciences, 2950 Hobson Way, Wright-Patterson AFB, 45433, OH, USA*

Abstract

Feature selection may be summarized as identifying salient features to a given response. Understanding which features affect the response enables, in the future, only collecting consequential data; hence, the feature selection algorithm may lead to saving effort spent collecting data, storage resources, as well as computational resources for making predictions. We propose a generalized approach to select the salient features of data sets. Our approach may also be applied to unsupervised datasets to understand which data streams provide unique information. We contend our approach identifies salient features robust to the subsequent predictive model applied. The proposed algorithm considers all provided variables, square variables, and two-way interactions as an extended data set. The algorithm implements a forward selection approach, based on correlation with the response, while fitting deep neural networks to the selected variables. These deep neural networks maintain an adaptive architecture which mirrors a full factorial design. These networks assess numeric and categorical values for both features and responses. Implementing this approach in ensemble with Recursive Feature Elimination we establish a new Pareto Frontier, consisting solely of this technique, for the Wisconsin Breast Cancer problem instance. This Pareto Frontier highlights our ensemble approach as the best performing method in both feature reduction and predictive accuracy.

Keywords: feature selection, neural networks, deep learning, machine learning interpretability, explainable artificial intelligence

1. Feature Selection

Data collection and storage capabilities have increased dramatically over the past 10 years. As a result, data scientist and researchers from many disciplines are over inundated with data. For machine learning applications, database records along with their associated field are processed into matrices of observations with associated features, which

are measurable characteristics of the observations. A vast number of features often poses a challenge when developing predictive models. Given a dataset with a large feature space, our goal in this study is to propose a technique that identifies the salient, or important, features related to a specified feature, deemed the response. This task is commonly referred to as Feature Selection (FS).

The benefits to accomplishing FS are threefold: we save time and resources by collecting less data in the future; we save memory by storing less data; we reduce computational complexity by analyzing less data. Besides these three advantages of a smaller data set, two potential application benefits include improving our predictive capability by reducing the "noise" in our data set, and easing the ability to understand relationships and explain insights from our data [1, 2].

Our article proposes and tests a generalized approach to feature selection. Sections are organized as follows: Previous Work reviews related articles. Evaluation Criteria describes our metrics for measuring success. State-of-the-Art identifies the current standard that we compare our algorithm against. Proposed Generalized FS Approach describes our proposed approach. Key FS Algorithm Differences highlights our algorithm's unique elements. Experiments and Results illustrates our method's performance in comparison to the current standard. Conclusions provides key takeaways.

2. Previous Work

Data scientists have been conducting research on improving FS methods over the past 28 years. Early methods focused on filter and screening techniques which can be summarized as a "leave-one-out" or "likelihood-ratio" approach [3, 4]. These early methods did not possess an adaptive model architecture, which would utilize ensemble search heuristics and prediction techniques dependent on problem size. Instead these early methods, commonly referred to as filter methods, mostly relied on regression techniques and parametric statistical tests to identify feature importance. Upon recognizing the benefits of adaptive model architectures, the FS community shifted towards wrapper and embedding techniques, which implement forward/backward search strategies and could be paired with a variety of predictive techniques. Deep Neural Networks (DNNs) were among the most popular predictive techniques [5, 6, 7, 8]. Steppe bridged the gap between the likelihood-ratio and DNN based methods [9]. In the last few years, FS research has focused on the Genetic Algorithm (GA) meta-heuristic's capabilities in ensemble with various prediction methods. While the capabilities of GA were recognized in early research, executing the algorithm has not become practical until recently [10, 2, 11, 12].

3. Evaluation Criteria

Past efforts have improved feature selection based on the number of retained fea-

tures, classification accuracy, and computational complexity. The Neural Information Processing Systems (NIPS) 2003 Feature Selection Challenge utilized these evaluation criteria and their influence carried into future work [13, 14]. These evaluation criteria base a FS method's success on the internal predictive modeling approach regardless of the successive modeling approach applied. A 2015 survey on feature selection concluded:

> "While there is no silver bullet method, filters based on information theory and wrappers based on greedy stepwise approaches seem to offer best results. Future research should focus on optimizing the efficiency and accuracy of feature subset search strategy by combining earlier best filter and wrapper approaches. Most research tends to focus on small number of datasets on which their methodology works." [12]

We strive with our proposed FS method to select appropriate salient features which are robust to the subsequently applied predictive model. To this goal, we contend comparing FS methods on the basis of retained features is only beneficial if we know the ground truth for the features that should be retained. Hence, we use generated, also referred to as artificial data with known salient features along with noise to test FS algorithms. This method has been proposed in previous studies [4, 15], however has not been fully adopted across the FS community [16]. We contend, that without knowing ground truth for influential features, attempting to select the fewest features based on an algorithm's predictive capability results in the best features for only that prediction technique. Selecting based on an internal prediction technique is probably not robust to use of other predictive methods which researchers may apply subsequently to the feature selection.

A wide variety of predictive models are used across FS methods. Each model is subject to it's own assumptions. For example, many of the FS methods in late 1990's and early 2000's were subject to the assumption that the relationship between the selected features and response is linear, and while newer methods, such as the two-layer genetic algorithm and elastic net proposed in 2021, search the feature space in a non-linear fashion, the final output from the elastic net is still subject to linearity [17, 18, 2]. This dependency on the prediction model is exacerbated by the fact that algorithm used for feature selection may not by applied by data scientists for their final predictions. Researchers from other domains most often want to implement a feature selection algorithm only to obtain salient features. They use the selected features as inputs to a separate domain-specific predictive model, which is independent of any model used in the feature selection algorithm [19, 20, 21, 22, 23]. We therefore contend that comparing performance across feature selection algorithms on the basis of their classification accuracy, or any other predictive capability metric, is not the best approach. Take for example, two hypothetical feature selection algorithms: al-

gorithms A and B. Let us suppose A and B both identify 3 salient features, specifically the same 3 salient features, except that A has a 10% better classification accuracy than B. We contend that the performance of these models is independent of classification accuracy since they identify the same features. In our generalized approach, we allow the researcher to input the selected features into any separate domain-specific predictive model. We acknowledge that predictive performance metrics are often appropriate factors for algorithm termination criteria (i.e. within a single algorithm), however we contend that these metrics should not be used to compare differing feature selection algorithms (i.e. across multiple algorithms). Instead we should evaluate FS algorithms based on their ability to identify a known set of salient features from a larger dataset.

While some researchers are concerned with the computational complexity or execution speed of FS algorithms, we contend the FS run-time is not a point of concern so long as its order of growth with respect to the number of features is less than exponential (complete enumeration) [16]. Our team assumes that researchers want to perform feature selection once (or rarely) in a program while they may be collecting data and calculating predictions on a recurring schedule. We imagine some entity concerned with implementing feature selection in the development of a particular predictive model to understand which features are important to their target response, and which data sources may be deprecated. The time savings offered across the lifetime of a project may outweigh extra time spent up front identifying salient features.

The main take away here is that we as a community are not doing our best job in evaluating FS techniques. We have focused too much on our predictive accuracy metrics, likely because that is what the larger scientific community is concerned with. However we must remember that our FS techniques exist only as a helping hand to the larger scientific community and are not intended to make final predictions.

To restate our main three points in this section:

1. We must have ground truth for salient features to evaluate performance across feature selection algorithms.
2. Our goal in developing an effective feature selection algorithm is independent of providing predictions as most researchers simply use the features identified by our algorithm in a domain specific predictive model.
3. Time spent implementing a good feature selection algorithm once up-front during a project saves time and money over the life of the project.

4. State-of-the-Art

For a current baseline for FS, we sought an algorithm that is public available and widely-used, therefore, we identified the state-of-the-art feature selection algorithm to be the Recursive Feature Elimination (RFE) method which exists in Python's Scikit-Learn module [20, 21, 23]. RFE is a backwards-selection

technique. RFE requires the user to define the desired number of features to be retained, the desired number of features to drop at each step, and a prediction method (Decision Tree, Random Forrest, etc.). Based on these inputs RFE executes a model with all features to obtain feature importance. It then prunes the desired number of features to drop from the model before re-executing to update feature importance metrics. This loop repeats until the desired number of features to be retained is met [24].

For best results with RFE the user should search a range for the number of retained features selecting the feature set with the best performance metrics. Due to RFE's dependence on the user to specify the number of features to retain, potential exists for the user to miss the appropriate number of salient features. Rather than constrain ourselves to a fixed number of retained features, our team proposes a method which inherently identifies the number of salient features without inputs from the user. In doing this, we strive to have the data direct us to the number of salient features rather than trying to direct our data to a pre-established number of desired features.

Another method we examine is the Boruta algorithm[15]. To identify the number of salient features in a dataset, Boruta introduces artificial noise to a dataset and examines relative feature importance inherent to it's random forest predictor. In 2020 Keany compared four methods for calculating feature importance in ensemble with Boruta and proposed the Boruta-Shap method which uses Shaply Additive Explanations. The four methods examined were the Boruta-Base implementation, Boruta-Permutation, Boruta-Gain (Gini), and Boruta-Shap. We refer the reader to Keany's work for full details on each method.[25] The Boruta-Shap method was quickly adopted.[22] While we contend Boruta should implement F-tests rather than t-tests as the t-distribution does not constitute a response surface, we acknowledge the merit in maintaining internal decision criteria when identifying the number of salient features [26].

5. Proposed Generalized FS Approach

5.1. Overview

Described at the highest level, our method is a feed-forward construction heuristic which evaluates all features through a Deep Neural Network (DNN). Our DNN inputs and architecture are inspired by a full factorial design from the Design of Experiments (DOE) research field. Figure 1 offers a flowchart for our proposed generalized FS algorithm. Our approach goes through Pre-Processing and Baseline Run stages before entering a Feature Addition Loop which includes stages: Add Feature Set, Initialize & Train New DNN, and Stop Check. Once we encounter any stopping criteria, we exit our feature addition loop and enter our last stage, Identify Salient Features, before terminating the algorithm. Our approach operates under the assumption the data has already been "cleaned" (not missing any values). Our proposed generalized method is capable of assessing features for datasets containing a continuous response, categorical response, or no response

(regression, classification, unsupervised). For unsupervised datasets, we find naturally occurring clusters in the data through the K-means++ clustering algorithm and use each points' cluster as a categorical response. This method for transforming an unsupervised dataset into a dataset containing a categorical response as it relates to feature selection has been implemented in more recent FS algorithms [2].

The remainder of this section is broken down into subsections covering our algorithm stages in order. Prior to Baseline Run, we include additional subsections, DNN Architecture & Attributes and Training Strategy, which describe the components of our deep neural networks. These additional sections are necessary to understand the setup of our DNN prior to examining the output from our DNN. We end this section by providing a comprehensive list and discussing Hyperparameters for our method.

5.2. Pre-Processing

During pre-processing, we identify the provided response or create a constructed response for unsupervised data. Our constructed response is built through K-means++ clustering. All categorical variables are one-hot encoded, which is done by converting a categorical feature into a sequence of binary variables. Our response and all features are individually standardized. Additionally we construct variables for all two-way interactions and quadratic terms for all provided features. It is worth noting that one-hot encoded features do not require

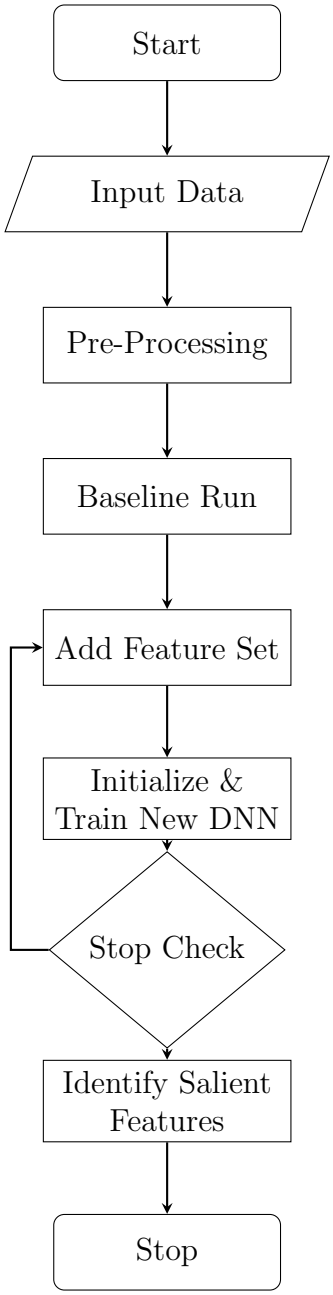

Figure 1: Algorithm Flow Chart

quadratic terms nor interactions across levels within the same feature. A binary variable could be used to create an interaction variable that is a select subset of the other variable. Next, our constructed interaction and quadratic terms are included in our feature list, and henceforth the term "features" refers inclusively to an expanded feature list which includes all originally provided features along with their constructed two-way interactions and quadratic terms. The set of originally provided features is referred to as our main effect features or simply main effects. We construct these additional terms and utilize them along with our main effects in our DNN based on findings in DOE that suggest the vast majority of all feature-response relationships can be captured using a combination of these terms. These DOE studies also suggest that even if higher order feature interaction terms are statistically significant to a response, their effect sizes are usually minimal compared to main effect, two-way interaction, and quadratic terms [27].

The next step in pre-processing is to construct a complete correlation matrix and create our candidate feature list, which is an ordered feature list based on each features' correlation with the response. We use this list sequence when adding features to our DNN in our feature addition loop. In the case that our response is categorical, we calculate the correlation for each feature across all levels in the response and retain the highest value observed. Specifically, our candidate feature list is provided in descending order by correlation with the response where continuous features use Pearson's correlation and one-hot encoded features use biserial correlation. Explicit notation for each type of correlation can be found in [28]. An analogy is Pearson's correlation may be thought of as a light on a rheostat dial that can be adjusted continuously, while biserial correlation is a light on a traditional switch, which may only be set to on or off. In each case the intensity of the light represents the correlation between the variables. We decided to use biserial correlation over the chi-square method as biserial correlation offers greater resolution into feature levels. The chi-square method provides correlation between the feature as a whole and the response, while the biserial method provides correlation between each feature level and the response. This enables our algorithm not only to provide the relevance of a categorical feature, but also the relevance of each level within a categorical feature.

The last step in our pre-processing stage is to format our data for our DNN. This includes: transforming the data into an array structure; vectorizing the response; and creating training/validation subsets from our data. We contend that a test subset is not required for this process as this algorithm is not used in making final predictions, only in identifying salient features which can be assessed through validation accuracy metrics. Continuous response problems maintain a response vector length of 1, while categorical response problems maintain a response vector length equal to the number of levels present in the response.

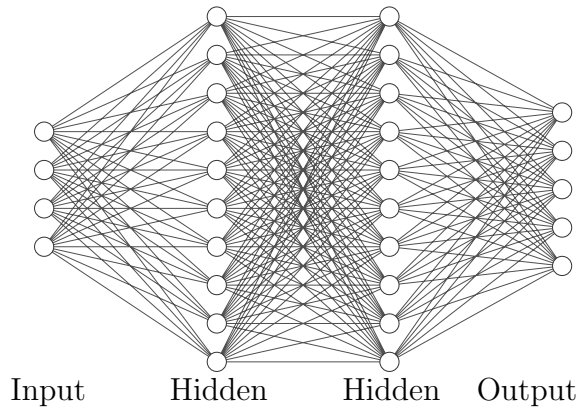

Figure 2: NN Architecture - 4 Inputs : 5 level Categorical Output [Figure developed with [29]]

5.3. DNN Architecture & Attributes

Our DNN architecture is defined to be the number of nodes in our input and hidden layers, which are determined by the number of currently selected features (N). As the algorithm selects additional features, this number (N) increases. Again, we are drawing inspiration from a full factorial designed experiment. Specifically our DNN has four layers: input (size: N), first hidden layer (size: $\binom{N}{2}$), second hidden layer (size: $\binom{N}{2}$), output (size: length of response vector). One exception to this framework is, in the case that N is less than 5, we default to using 10 nodes in our hidden layers. The number of nodes in the output layer equals the length of our response vector. Figure 2 shows an example architecture for a hypothetical iteration of our algorithm which contains 4 features and a categorical response with 5 levels.

Our DNN attributes are defined to be the activation functions used in each layer, loss metric, accuracy metric, optimizer, and learning rate. These attributes are common DNN hyper-parameters and the detailed function of each is covered by Géron and Aurélien [30]. For our proposed method, the DNN attributes vary by problem type: continuous versus categorical responses. Table 1 provides the attributes we used for each of the two problem types. While we use a linear activation function in continuous problems, because we constructed the non-linear interaction and quadratic terms, we are still able to capture these non-linear relationships. Additional non-linear transformations could be considered for future research, however based on the DOE insights previously mentioned and our DNN depth, we are confident the proposed method captures the vast majority of feature-response relationships.

Table 1: NN Attributes

Attribute	Continuous	Categorical
Input and hidden activations	Relu	Relu
Output activation	Linear	Softmax
Loss	Mean Squared Error	Categorical Cross Entropy
Accuracy	Mean Squared Error	Accuracy
Optimizer	Stochastic Gradient Descent	Stochastic Gradient Descent
Learning rate	0.01	0.01

5.4. Training Strategy

Each iteration within our algorithm, or run, trains with a unique feature set for a fixed number of epochs (ϵ). Each run begins training from scratch (randomized weights). This strategy treats each new feature set as a new problem and due to our network's adaptive architecture, the network's ability to process information grows with respect to the amount of information provided

while our minimum node requirement protects against missing information when few features are used. We contend this training strategy is equally capable of exploiting relevant information (salient features) at any point throughout the algorithm and possesses no bias towards early/late feature additions.

For a given run, after performing the prescribed number of epochs, we average the validation accuracy metric of the best (δ) epochs from that particular run. This averaged validation accuracy metric is considered our current feature set's performance metric. The intent behind using an averaged performance metric for a feature set is to avoid bias from a single uncharacteristically high or low performing epoch which may result due to the DNN's stochastic nature. It is important to note that in regression (continuous response) problems a lower MSE represents better performance, while in classification (categorical response) problems a higher accuracy represents better performance. Additionally we always consider the validation metric rather than training to avoid models which suffer from overfitting.

5.5. Baseline Run

We obtain a baseline performance metric by training our model using all main effects from our feature space. This baseline performance metric influences our stopping criteria. This method utilizes neural networks' ability to make accurate predictions with "noisy" data [31]. While noise does not impede good predictions, it does impose additional limits on interpretability and explainability where most DNNs are already considered a "black-box" technique. Linardatos, Papastefanopoulos, and Kotsiantis provide great review and discussion of what constitutes explainable artificial intelligence (XAI), and the difference between interpretability and explainability [32]. Often these are considered together under the field of XAI, however to be more specific interpretability focuses on why we receive a particular output while explainability focuses on how we receive a particular output. By using a baseline performance metric created with all main effects, identifying salient features offers our DNN interpretability.

5.6. Feature Addition Loop

This section covers three stages in our algorithm: adding new feature sets from our candidate feature list to our model, initializing and training a new DNN from scratch (no previous weights carried forward), and checking our current run's performance against three stopping criteria.

When adding features to our model we again are influenced by DOE through a concept called model hierarchy. Model hierarchy maintains that if we include a higher order effect in our model (interaction or quadratic term) then we must also include it's corresponding main effects [27]. For example if we decide to train our DNN with the interaction term for $X_1 * X_2$ then we must also include the individual X_1 and X_2 main effects in our training set. Additionally, to avoid adding redundant information, our algorithm only adds a feature to our model if it's correlation with all features currently in the model is below a user defined addition

correlation threshold (ξ). To avoid adding redundant features, this threshold is prioritized over model hierarchy in our algorithm.

We continue adding feature sets sequentially from our candidate feature list and training a new DNN each run until we reach our prescribed stopping criteria. When any stopping criterion is encountered, the feature addition loop terminates and we identify salient features before exiting the algorithm. When discussing stopping criteria, it is important to recognize that independent values must be provided for both regression and classification problems since they maintain different accuracy metrics. Our algorithm maintains three stopping criteria based on a given run's performance metric:

- Sufficient Performance
- Baseline-Relative Performance
- Improved-Degraded Performance

The first stopping criteria, sufficient performance, is purely based on a user determined "good enough" solution. This value is measured by respective accuracy metrics, accuracy or MSE, and is represented by (α). The second stopping criteria, baseline-relative performance, is based on a user defined baseline-relative performance threshold (ϕ). This threshold is set to be proportional to our baseline performance metric. To exemplify this consider a classification problem where our baseline performance equals 95% accuracy and ϕ equals 0.9, then our baseline-relative performance stopping criteria would be achieving an accuracy of 85.5% $= (0.95 * 0.9)$. Conversely this example may be thought of as accepting a drop in 10% of our baseline accuracy. This converse approach is used in defining the metric for the continuous case. Consider a continuous response problem where our baseline performance equals 10 MSE (standardized response units) and ϕ equals 1.1, then our baseline-relative performance stopping criteria would be achieving a MSE of $11 = (10 * 1.1)$. The third stopping criteria, improved-degraded performance, only occurs if we have improved our performance metric since our baseline. This translates to improving performance while reducing our feature space. In this case, we continue adding features as long as we continue improving performance, and once a new feature results in degraded performance from the previous set, we revert back to the last feature set that increased our performance by some threshold (ω). We do not simply revert back to the most recent feature set as DNN's perform well when making predictions with "noisy" data: In this specific case, it is possible to have added features over multiple runs with negligible improvement due to the stochastic nature of the DNN.

5.7. Identify Salient Features

Upon exiting the feature addition loop, all main effects in our model are identified as salient. We intentionally avoid providing the additional terms we've constructed (squared and interaction) as salient features in order to maintain a robust set and avoid influencing the user in their subsequent analysis. For example our algorithm may recognize X as

salient through it's quadratic term while the true relationship was exponential. This is in alignment with our philosophy that FS algorithms should not be used to make final predictions and researchers using our algorithm likely maintain separate domain specific predictive models and may be using this tool to understand which data streams may be deprecated: this tool is designed to provide a robust salient feature list from the original set of salient features which characterize our feature-response relationship rather than offer final predictions or suspected inter-feature relationships. This determination is further supported by the Universal Approximation Theorem which concludes that even the most basic neural networks are capable of approximating any non-linear function to some degree of accuracy [33]. Consider a case in which we use an interaction term in training our DNN, the user may miss this interaction if they subsequently apply simple linear regression, however they are likely catch the interaction if they subsequently apply a tree based method. We acknowledge that identifying inter-feature relationships is paramount in conducting thorough analysis, however we contend these relationships' expression is dependent on the predictive method applied and therefore should be left to analysis post feature selection. This intent applies to relative feature importance as well. Take again our example of FS methods A and B which identify the same 3 salient features: the relative feature importance of each variable may vary between the methods dependent on their respective predictive components.

5.8. Hyper-parameters

To conclude this section we identify, and discusses sensitivity for, hyper-parameters within our algorithm. Hyper-parameters include:

- NN Attributes: Reference Table 1
- Minimum Nodes (η)
- Epochs (ϵ)
- Number Epochs Averaged (δ)
- Addition Correlation Threshold (ξ)
- Sufficient Threshold (α)
- Baseline-Relative Threshold (ϕ)
- Improvement-Degraded Threshold (ω)

While the NN attributes identified in table 1 are commonly adjusted when optimizing a neural network for predictions, these are not designed to be adjusted as part of our method. These specific attributes were selected based on their unique capabilities and alignment with the DOE principals leveraged. The user may however define a minimum number of nodes (η) to be used in the hidden layers. This number is used for both layers, and is analogous to information processing capability.

The number of epochs, the number of iterations through all provided data points per run, is represented by ϵ. This number is analogous to the amount of effort we want our DNN to provide when extracting information form a given feature set. This number remains constant for all runs. Setting this

number too low results in missed information while setting it too high results in unnecessarily long run times. We are not concerned with overfitting if the number of epochs is set too high as we only ever consider the validation accuracy metrics. Additionally the user may define a number of top-performing epochs (δ) to be averaged. Epoch performance is compared by the validation accuracy metric. The calculated average is used as the run's performance. Setting this δ too low may result in capturing uncharacteristically high performance, while setting it too high may result in capturing uncharacteristically low performance. The user may also define the feature addition correlation threshold (ξ), which must range from 0 to 1, and is analogous to limiting redundant information. Setting ξ too low may result in capturing redundant features, while setting it too high may result in missing salient features.

The final three hyper-parameters are related to our stopping criteria. Again for these criteria it is important to note that independent values must be provided for regression and classification problems as their performance metrics are inversely related (i.e. a lower MSE is better while higher accuracy is better). Alpha (α) represents our sufficient performance threshold. Setting this threshold to an extremely high performance level simply makes us more likely to terminate based on one of the other criteria. On the other hand, setting this threshold to an extremely low performance level is likely to prematurely terminate our algorithm and miss salient features. Phi (ϕ) represents our baseline-relative performance threshold. This is analogous to a tolerable trade-off between predictive power and the amount of information needed to make predictions. This is accomplished by scaling accuracy down for classification problems and scaling MSE up for regression problems. Allowing too large a deficit from our baseline performance will result in missing salient features and likely poor predictions in subsequent analysis. If we set ϕ equal to 1, this represents the case in which we have zero tolerance for reduced performance from our baseline and we only reduce our feature space if doing so improves our performance or the performance is still within our sufficiency threshold. Omega (ω) represents our improvement-degraded threshold. This hyper-parameter only affects our algorithm in the case that we improve performance from our baseline by reducing our feature space. This value balances the greedy nature in which we add features to our model. In the case that ω affects our algorithm, setting it too low may result in capturing redundant features, while setting it too high may result in missing salient features.

6. Key FS Algorithm Differences

Previous neural network based feature selection routines suffered from epoch dependency in which the methods were biased towards selecting new salient features in early runs [5]. This behavior is mirrored in meta-heuristic search based routines as well [2]. Traditionally, this is beneficial and described as balancing exploration and exploitation within a search space [34]. This exploration-exploitation trade off is not inherently part of

our algorithm, and as a result we do not bias our algorithm towards selecting features in earlier runs. We accomplish this through our feature addition criteria and training strategy for our DNN. With regards to exploration, while our DNN maintains stochastic elements, namely randomized weights and stochastic gradient descent, no random exploration exists in our algorithm while searching the feature space. Rather our proposed method's search strategy is directed based on DOE concepts of considering features based on main, interaction, and quadratic effects. Furthermore, we calculated these features' correlations with the response to guide our selection order. Considering exploitation, we contend our adaptive model architecture and training strategy assess each run independently, and each run is equally capable of exploiting information relative to the amount of information provided.

We contend that previous efforts, summarized in [16, 12], primarily address the question: How do we best make predictions while limiting our feature space? In contrast, we aim to answer the question: Given a list of variables which are important to retain and which may be deprecated? Our answer to this question is designed to be robust to subsequent predictive methods which may then begin to characterize inter-feature relationships and relative feature importance metrics. To this goal, we must also be cautious to not bias our results to our own DNN's predictive capabilities. Therefore we only identify main effects as salient and avoid characterizing inter-feature relationships as these characterizations are subject to the predictive method applied. In our algorithm, these inter-feature characterizations are considered to be: inter-feature correlation, constructed interaction and squared terms, as well as relative feature importance. If analysts desire to obtain feature importance from our model, which we advocate against, the user may calculate percentage improvement from the baseline mean squared error (MSE) or the raw accuracy improvement for a given run. An issue here is that our method maintains model hierarchy, therefore these feature importance values are confounded for higher order terms if it's main effects were not already in the training set. If we were to remove the model hierarchy requirement, we could directly capture individual feature importance values, however we do not find this trade off beneficial as the feature importance values would still be subjective to our predictive approach. If the user requires relative feature importance measures, we recommend implementing a tree or forest based prediction method post feature selection, or using an alternative feature selector which is structured around tree based methods [21, 25].

7. Experiments and Results

7.1. Overview

This section contains two experimental designs, and will refer to our proposed Generalized Robust Feature Selection method as GRFS in tables plots and figures. Table 2 outlines the default hyper-parameters our algorithm uses in each experiment unless otherwise noted. We recognize our newly proposed evaluation criteria based on artificial

data is not common to previous work. To offer a direct comparison to previous studies, the Breast Cancer Data section uses the NIPS 2003 Feature Selection Challenge evaluation criteria. NIPS prescribes the goal of FS is to maintain the highest accuracy possible with the fewest number of features [13]. Hence, higher and to the left is better in Figures 3 and 4. Comparison is made on the commonly known Wisconsin Breast Cancer Dataset maintained by the University of California Irvine Machine Learning Repository (UCI ML Repo) [35]. The Artificial Data section examines 14 artificial problem instances using our proposed evaluation criteria. In developing these problem instances, one hundred continuous and categorical features are each uniformly generated using Python's Numpy module maintaining unique random number streams to mitigate auto-correlation [36]. Categorical features are balanced across feature levels. Table 3 provides a detailed description of the 14 generated datasets and their respective response functions while Table 4 compares our method's performance to RFE on these datasets using our proposed evaluation criteria.

7.2. Breast Cancer Data

When proposing Boruta-Shap in 2020, Keany [25] provides four Boruta methods' performance on this common dataset. Keany cites this dataset as containing 32 features however one of these is the response and another is the row identification. Therefore, we consider the dataset to contain 30 features and adjust Keany's results accordingly by subtracting 2 from the number of retained features provided by Keany [25]. To match Keany's analysis, all accuracy metrics provided in this section are floor rounded.

Table 2: Hyper-parameter Settings

Hyper-parameter	Continuous	Categorical
NN Attributes	Table 1	Table 1
Minimum Nodes (η)	5	5
Epochs (ϵ)	10	10
Number Epochs Averaged (δ)	3	3
Addition Correlation Threshold (ξ)	0.8	0.8
Sufficient Threshold (α)	0.02	0.95
Baseline-Relative Threshold (ϕ)	1.1	0.9
Improvement-Degraded Threshold (ω)	0.01	0.01

We examine our algorithm's performance training for 15 epochs (ϵ) each run across three hyper-parameter settings representing different stopping criteria. Specifically we examine the effect of adjusting our sufficient and baseline-relative stopping criteria, α and ϕ respectively. Note that while our baseline performance influences our stopping criteria, our baseline run is a "stand-alone" run and is not affected by our stopping criteria. With this in mind, we first used our default hyper-parameter settings outlined in Table 2. Our baseline run, which retains all 30 features, achieves 98% accuracy. Using these default hyper-parameter settings, we exit our feature addition loop based on our baseline-relative stopping criteria (ϕ), and retain 6 features while achieving 92% accuracy. We then adjusted our stopping criteria in an attempt to match the best performing Boruta method, Boruta-Shap, which retains 19 features while achieving 96% accuracy. Our baseline per-

formance of 98% accuracy indicates we are capable of matching this performance. To achieve this, we set our baseline-relative performance stopping criteria (ϕ) to 1.0 and our sufficient performance threshold (α) to 96% (0.96). Remember setting ϕ to 1.0 effectively removes our baseline-relative stopping criteria and continues to add feature sets until we either reach our sufficiency threshold (α) or experience improved accuracy from our baseline then degrade by adding additional features which is not expected in this case. Using $\alpha = 0.96$ and $\phi = 1.0$, and leaving all other hyper-parameters unchanged from their defaults, our method retains 8 features while achieving 96% accuracy. Lastly, we were interested in matching our baseline accuracy in as few features as possible to offer full interpretability to our DNN. To achieve this we set $\alpha = 0.98$ and $\phi = 1.0$ leaving all other hyper-parameters unchanged from their defaults. With these values, our method retains 9 features while achieving 98% accuracy.

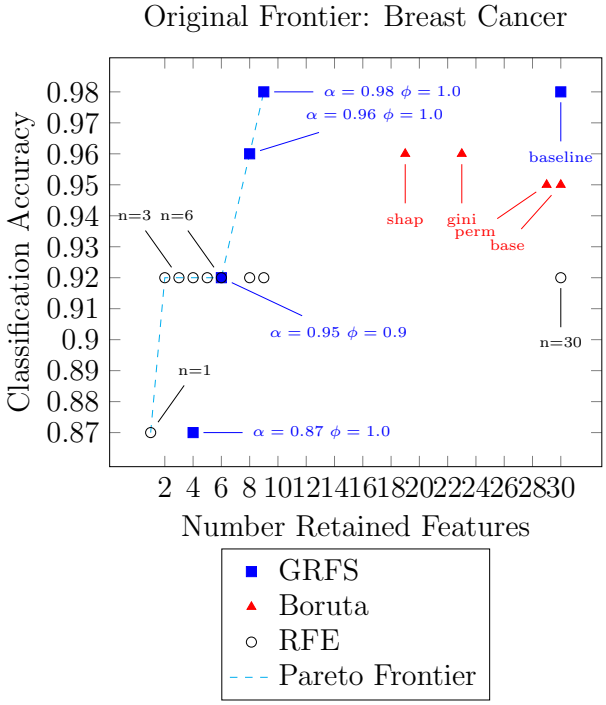

Figure 3: This Pareto Frontier consists of multiple methods therefore it is impossible to indicate a clear "winner". The best performing method in this case is subject to the users objectives.

As there is no ground-truth for the number of salient features in this dataset, we provide RFE the same number of features retained in each of our algorithm's runs and compare. We let n represent the number of features to be retained by RFE. RFE maintains 92% accuracy while decreasing the number of retained features from 30 to 2. Understanding higher and to the left is better, we construct a Pareto Frontier [37] in Figure 3. As multiple methods exist on this frontier, there is no clear "winner". However, recognizing the potential in each of these methods we developed an ensemble approach initializing our Generalized Robust Feature Selection algorithm with the output from RFE. We refer to this ensemble method as RFE-GRFS. Each time we use RFE's output as a starting point, our algorithm further reduces the feature space while improving or maintaining accuracy. Figure 4 illustrates this technique is the clear "winner" as it results in an entirely new Pareto Frontier which is made up solely by solutions from this ensemble technique. While the various Boruta methods maintain high accuracy and internal processes to determine the number of retained features, Boruta methods retain far more features.

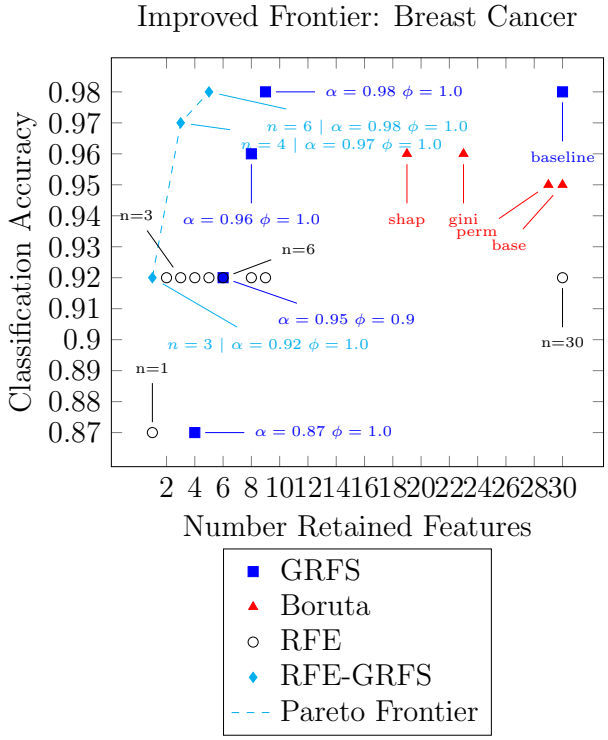

Figure 4: RFE-GRFS outperformed all other techniques and is now the only technique on our Pareto Curve. Therefore it is clearly the best performer for both objectives: It maintains the smallest feature space and the highest accuracy.

7.3. Artificial Data

When comparing our method to RFE in the artificial data experiments, we made the assumption that the user would include the true number of salient features in their search space when using RFE, and furthermore that this value would be selected based on RFE's internal performance metric. This assumption, while not characteristic of the effort required by the user, removes subjectivity based on user input and focuses on the internal algorithm capabilities. Additionally

Table 3: Artificial Data Description - Type represents Regression (R) and Classification (C)

Experiment	**Type**	**Description**	**Response (Y)**
High Correlation	R	4 repeated features with exact correlation (Pearsons = 1.0)	$Y = X_1 + X_5 + X_6$ Where: $X_1 = X_2 = X_3 = X_4$
Moderate Correlation	R	10 features constructed as pairwise sums of initial 20 (max Pearsons = 0.73)	$Y = X_1 + X_{15}$ Where: $X_{21} = X_1 + X_{11}$ $X_{22} = X_2 + X_{12}$ . . . $X_{30} = X_{10} + X_{20}$
100 Features	R	large feature space	$Y = \sum_{i=1}^{20} X_i$
Single Quadratic	R	response is quadratic of main effect	$Y = X^2$
Three Quadratics	R	response is sum of three quadratics	$Y = X_1^2 + X_2^2 + X_3^2$
Single Two Way Interation	R	response is a single two way interaction	$Y = X_1 * X_2$
Five Two Way Interactions	R	response is sum of five two way interactions	$Y = (X_1 * X_2) + (X_3 * X_4) + (X_5 * X_6) + (X_7 * X_8) + (X_9 * X_10)$
Simple Binary Feature	R	continuous response based on binary condition	*If* Binary = 1: $Y = X_1$ *Else*: $Y = X_2$
Complex Binary Feature	R	continuous response based on binary condition	*If* Binary = 1: $Y = X_1 + X_2$ *Else*: $Y = X_3 * X_4$
Binary Response	C	binary response based on continuous conditon	*If* $X_1 + X_2 > 1000$: $Y = 1$ *Else*: $Y = 0$
Periodic Response	R	representative of trigonometric, harmoinc, tim...	$Y = \sin X$
Exponential Response	R	response is exponential of main effect	$Y = \exp X$
Logistic Growth Response	R	response is logistic growth (sigmoid) using main effect	$Y = \frac{1000}{(1+\exp(-0.01*(X-500)))}$
Multi-Categorical Response	C	multi-categorical response based on continuous condition	*If* $X <= 200$: $Y = Type_1$ *Elif* $200 < X <= 400$: $Y = Type_2$ . . . *Elif* $800 < X$: $Y = Type_5$

we one-hot encoded categorical features before passing them to RFE. Our RFE instantiation utilized the Decision Tree Regressor and Decision Tree Classifier predictive models, which exists as part of Python's Scikit-

Learn module, for regression and classification problems respectively [24]. In developing and executing our algorithm: we established our DNN attributes and architecture using Python's Keras module, and trained our DNN using Python's TensorFlow backend. [38, 39] Our experiments were executed in Python 3.7.6 [40].

Table 4 provides the number of true salient and noise features for each experiment along with the number of correctly identified salient features and incorrectly retained noise features for both our method and RFE. Each method was 100% accurate, identifying all salient features and removing all noise, in 11 of the 14 experiments. The three problem instances where we observed discrepancies include: 100 Features, Complex Binary Feature, and Periodic Response. Descriptions of these instances and their respective response functions are provided in Table 3. In our 100 Feature problem instance, our method performed with 100% accuracy correctly identifying all salient and removing all noise features while RFE correctly identified 18 of 20 salient features and incorrectly retained 2 of 80 noise features. In our Complex Binary Feature problem instance, our method correctly identified 3 of 5 salient features and removed all noise features while RFE correctly identified 3 of 5 salient features and incorrectly retained 2 of 5 noise features. In our Periodic Response problem instance, both our method and RFE fail to identify the salient feature used to generate the sine curve. In this instance our method incorrectly retains 2 of 10 noise features while RFE incorrectly retains 1 noise feature. As RFE fails to individually identify all salient features in our largest problem instance, we did not find it beneficial to examine our RFE-GRFS ensemble approach against these artificial data. In these artificial data experiments, autonomy, specifically interpretable autonomy, is our algorithm's primary benefit when compared to RFE.

Table 4: Artificial Data Results - True Salient Features (TS), True Noise Features (TN), Identfied Salient Features (Method-S), Retained Noise Features (Method-RN)

Experiment	TS	TN	GRFS-S	GRFS-RN	RFE-S	RFE-RN
High Correlation	3	7	3	0	3	0
Moderate Correlation	2	28	2	0	2	0
100 Features	20	80	20	0	18	2
Single Quadratic	1	9	1	0	1	0
Three Quadratics	3	7	3	0	3	0
Single Two Way Interation	2	8	2	0	2	0
Five Two Way Interactions	10	40	10	0	10	0
Simple Binary Feature	3	7	3	0	3	0
Complex Binary Feature	5	5	3	0	3	2
Binary Response	2	9	2	0	2	0
Periodic Response	1	9	0	2	0	1
Exponential Response	1	9	1	0	1	0
Logistic Growth Response	1	9	1	0	1	0
Multi-Categorical Response	1	10	1	0	1	0

8. Conclusions

As suspected by Jović, Brkić, and Bogunović [12], best performance was achieved using an ensemble filter and wrapper technique. While Boruta-Shap offers improvement to the baseline Boruta algorithm, none of the four cited Boruta methods provided suitable performance in feature reduction or predictive accuracy. Our ensemble technique, utilizing Recursive Feature Elimination as a filter for our Generalized Robust Feature Selection algorithm, provided superior performance in both feature reduction and predic-

tive accuracy. This performance is illustrated as a Pareto Frontier constructed solely from this technique's solutions [Figure 4]. Considered individually on artificial data, our algorithm performs better in identifying salient features when compared to Recursive Feature Elimination.

References

[1] Z. Chen, Q. Chen, Y. Zhang, L. Zhou, J. Jiang, C. Wu, Z. Huang, Clustering-based feature subset selection with analysis on the redundancy–complementarity dimension, Computer Communications 168 (2021) 65–74.

[2] F. Amini, G. Hu, A two-layer feature selection method using genetic algorithm and elastic net, Expert Systems with Applications 166 (2021) 114072.

[3] M. A. Hall, Correlation-based feature selection of discrete and numeric class machine learning (2000).

[4] H. Liu, R. Setiono, et al., A probabilistic approach to feature selection-a filter solution, in: ICML, Vol. 96, Citeseer, 1996, pp. 319–327.

[5] M. M. Kabir, M. M. Islam, K. Murase, A new wrapper feature selection approach using neural network, Neurocomputing 73 (16-18) (2010) 3273–3283.

[6] K. L. Moore, Salient feature selection using feed-forward neural networks and signal-to-noise ratios with a focus toward network threat detection and classification, Tech. rep., AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT-PATTERSON AFB OH GRADUATE SCHOOL OF ... (2014).

[7] R. Setiono, H. Liu, Neural-network feature selector, IEEE transactions on neural networks 8 (3) (1997) 654–662.

[8] A. Verikas, M. Bacauskiene, Feature selection with neural networks, Pattern recognition letters 23 (11) (2002) 1323–1335.

[9] J. M. Steppe, Feature and model selection in feedforward neural networks, Tech. rep., AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING (1994).

[10] R. Leardi, A. L. Gonzalez, Genetic algorithms applied to feature selection in pls regression: how and when to use them, Chemometrics and intelligent laboratory systems 41 (2) (1998) 195–207.

[11] T. HO, D. W. COWAN, P. M. McLendon, H. A. Pangburn, Hybrid feature selection with genetic algorithms and other methods, Tech. rep., Rensselaer Polytechnic Institute (2020).

[12] A. Jović, K. Brkić, N. Bogunović, A review of feature selection methods with applications, in: 2015 38th international convention on information and communication technology, electronics and mi-

croelectronics (MIPRO), Ieee, 2015, pp. 1200–1205.

[13] I. Guyon, S. Gunn, M. Nikravesh, L. A. Zadeh, Feature extraction: foundations and applications, Vol. 207, Springer, 2008.

[14] I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, M. Uhr, Feature selection with the clop package, Technical Report (2006).

[15] M. B. Kursa, W. R. Rudnicki, et al., Feature selection with the boruta package, J Stat Softw 36 (11) (2010) 1–13.

[16] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[17] R. Porkodi, Comparison of filter based feature selection algorithms: An overview, International journal of Innovative Research in Technology & Science 2 (2) (2014) 108–113.

[18] D. Garrett, D. A. Peterson, C. W. Anderson, M. H. Thaut, Comparison of linear, nonlinear, and feature selection methods for eeg signal classification, IEEE Transactions on neural systems and rehabilitation engineering 11 (2) (2003) 141–144.

[19] F. Mobley, A. T. Wall, H. Gallagher, Curve-fit of spectral variations of noise in f-35a cockpit, in: Proceedings of Meetings on Acoustics 178ASA, Vol. 39, Acoustical Society of America, 2019, p. 045012.

[20] K. Yan, D. Zhang, Feature selection and analysis on correlated gas sensor data with recursive feature elimination, Sensors and Actuators B: Chemical 212 (2015) 353–363.

[21] B. F. Darst, K. C. Malecki, C. D. Engelman, Using recursive feature elimination in random forest to account for correlated variables in high dimensional data, BMC genetics 19 (1) (2018) 1–6.

[22] M. J. Kleiman, E. Barenholtz, J. E. Galvin, A. D. N. Initiative, et al., Screening for early-stage alzheimer's disease using optimized feature sets and machine learning, Journal of Alzheimer's Disease 81 (1) (2021) 355–366.

[23] F. Mobley, A. Wall, C. Campbell, Selection of Salient Aircraft Parameters for Modeling Acoustic Levels within Fifth Generation Aircraft (2022).

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

[25] E. Keany, BorutaShap : A wrapper feature selection method which combines the Boruta feature selection algorithm with Shapley values. (Nov. 2020).

doi:10.5281/zenodo.4247618.
URL https://doi.org/10.5281/zenodo.4247618

[26] L. J. Bain, M. Engelhardt, Introduction to probability and mathematical statistics, Vol. 4, Duxbury Press Belmont, CA, 1992.

[27] D. C. Montgomery, Design and Analysis of Experiments, Wiley, 2012.

[28] R. F. Tate, The theory of correlation between two continuous variables when one is dichotomized, Biometrika 42 (1/2) (1955) 205–216.

[29] A. LeNail, Nn-svg: Publication-ready neural network architecture schematics., J. Open Source Softw. 4 (33) (2019) 747.

[30] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems, O'Reilly Media, 2019.

[31] D. Rolnick, A. Veit, S. Belongie, N. Shavit, Deep learning is robust to massive label noise, arXiv preprint arXiv:1705.10694 (2017).

[32] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, Explainable ai: A review of machine learning interpretability methods, Entropy 23 (1) (2021) 18.

[33] J. Bill, L. Champagne, B. Cox, T. Bihl, Meta-heuristic optimization methods for quaternion-valued neural networks, Mathematics 9 (9) (2021) 938.

[34] J. G. March, Exploration and exploitation in organizational learning, Organization science 2 (1) (1991) 71–87.

[35] D. Dua, C. Graff, UCI machine learning repository (2017).
URL http://archive.ics.uci.edu/ml

[36] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362. doi:10.1038/s41586-020-2649-2.
URL https://doi.org/10.1038/s41586-020-2649-2

[37] P. Ngatchou, A. Zarei, A. El-Sharkawi, Pareto multi objective optimization, in: Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems, IEEE, 2005, pp. 84–91.

[38] F. Chollet, et al., Keras (2015).
URL https://github.com/fchollet/keras

[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp,

G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL https://www.tensorflow.org/

[40] G. Van Rossum, F. L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2022	Master's Thesis	Sept 2020 — Mar 2022

4. TITLE AND SUBTITLE

Generalized Robust Feature Selection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lott, Bradford, 1st Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-22-M-148

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Feature selection may be summarized as identifying salient features to a given response. We propose a generalized approach to select the salient features of data sets. Our approach may also be applied to unsupervised datasets to understand which data streams provide unique information. We contend our approach identifies salient features robust to the subsequent predictive model applied. The proposed algorithm considers all provided variables, square variables, and two-way interactions as an extended data set. The algorithm implements a forward selection approach, based on correlation with the response, while fitting deep neural networks to the selected variables. These deep neural networks maintain an adaptive architecture which mirrors a full factorial design. These networks assess numeric and categorical values for both features and responses. Implementing this approach in ensemble with Recursive Feature Elimination we establish a new Pareto Frontier for the Wisconsin Breast Cancer problem instance. This Pareto Frontier highlights our ensemble approach as the best performing method in both feature reduction and predictive accuracy.

15. SUBJECT TERMS

feature selection, neural networks, deep learning, machine learning interpretability, explainable artificial intelligence

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Mark A. Gallagher, AFIT/ENS
U	U	U	UU	26	19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565; Mark.Gallagher@afit.edu