Using SHAP Post-Model Explainability as a Model-Agnostic Feature Selection Technique

Joshua Gottlieb
Seidenberg School of CSIS
Pace University
New York City, USA
jg05394n@pace.edu

Martin Sichali Chunsen
Seidenberg School of CSIS
Pace University
New York City, USA
cs83339n@pace.edu

Alexander Yoo
Seidenberg School of CSIS
Pace University
New York City, USA
ay71173n@pace.edu

Advisor: Dr. Krishna Bathula
Seidenberg School of CSIS
Pace University
New York City, USA
kbathula@pace.edu

***Abstract*—Feature selection techniques are an important tool in the data science process, as they help reduce dimensionality of datasets, remove sparsity, and help eliminate noisy patterns within data, improving generalization of models to future data. While there are many feature selection techniques, many of these techniques either fail to optimize for the predictive power of the model, require iterating through the extensive feature space, or are applicable only to certain model types. We propose using SHapley Additive eXplanation (SHAP) values to capture and rank the important features learned by a model for use in feature selection due to their model-agnostic nature and their inherent focus on capturing the predictive power of each feature in the context of the particular model being evaluated. We build upon previous work using SHAP as a feature selection method by evaluating the effectiveness of using SHAP for feature selection across five different model types and ten diverse datasets. We propose a unique set of strategies for choosing the ranked features based on the combined and individual strengths of each feature as discovered by SHAP.**

***Index Terms*—Feature selection, model-agnostic methods, Shapley values, machine learning.**

I. Introduction

High dimensionality in datasets is a common occurrence and presents many issues. Not all features are relevant to the machine learning task, and higher dimensionality leads to an increase in noisy and erroneous patterns within the dataset. Data also becomes sparse in higher dimensions, requiring increasingly more data to achieve acceptable performance as dimensionality grows [1]. These issues lead to a decrease in model performance, as models fit themselves to irrelevant patterns and generalize poorly to new data, underperforming due to high dataset complexity.

There are projective techniques, such as PCA [2], t-SNE [3], or UMAP [4], which take high dimensional data and directly project it into a lower dimensional space. However, these dimensionality techniques have the unfortunate consequence of distorting or destroying all model interpretability due to combining features together in ways which obscure the effects of each contributing feature. This outcome is not ideal for most modeling scenarios, where it is necessary not only that a model works but to be able to understand why the model works and to pinpoint which features are most useful for the real-world task being modeled.

Feature selection is the alternative to dimensionality reduction techniques and works by selecting a subset of the features instead of using the entire feature set. By using each feature in its original form, model interpretability is preserved while reducing data dimensionality. In order to select features, a selection criteria (i.e., a metric) [5] is needed to rank features and find the optimal subset. In general, it is not feasible to test every possible feature subset, as the number of feature subsets for a dataset with n features is 2^n.

Existing feature selection techniques are divided into three groups: filter, wrapper, and embedded [5], [6]. Each of these groups has drawbacks when it comes to how they select feature subsets. Filter methods are unable to utilize the patterns learned by a model to select features; wrapper methods are computationally expensive and require a complex search strategy to test feature subsets; and embedded methods are model-specific and cannot be used across models. Instead of traditional feature selection techniques, it is possible to use a technique from the field of model explainability known as SHAP [7]. SHAP is a model-agnostic technique which attempts to explain black box models by assigning a contribution value for each feature at each data point. These SHAP values can be used as a proxy for feature importance and are the basis of the work produced in [8].

Our paper expands upon prior work using SHAP values for feature selection by testing a variety of models to assess the effectiveness of SHAP selection as a model-agnostic feature selection technique. These experiments are replicated across ten publicly available datasets to ensure the results are reproducible under different conditions and are not the result of particularly favorable conditions inherent to any single dataset. We also propose a unique set of strategies, grounded in human intuition, for choosing the k features to select during the SHAP selection process.

The remainder of this paper is organized as follows. Section II presents an overview of feature selection methods and of SHAP explanation algorithms. Section III briefly covers previous work using SHAP values for feature selection and identifies the gaps and drawbacks present in prior work that our work aims to address. Section IV outlines the methodology of our experiments. Section V contains the analysis and results of our experiments, while Section VI provides a succinct

conclusion of our findings and suggestions for further work.

II. Technical Background

A. Feature Selection Methods

The goal of feature selection methods is to find the ideal subset of features that are the most useful and relevant to the chosen machine learning task. Features may be relevant, irrelevant, or redundant [6], and different feature selection techniques tackle the problem of choosing only the relevant features in different manners. For a dataset with n features, the total possible number of feature subsets is 2^n, making it infeasible to test all possible feature subsets to find the optimal feature subset for most datasets. The ideal feature selection technique chooses only the best and most relevant subset of features; however, in practice, there is a trade-off between the efficiency and simplicity of the feature selection technique and its effectiveness in finding the best feature subset [1]. There are three main categories of feature selection methods: filter, wrapper, and embedded methods [5], [6].

Filter methods employ a performance measure in order to rank features by importance and relevance. Filter methods are not reliant on the final algorithm used for modeling and as such are model-agnostic. Filter methods can be further broken into categories based on their scoring criteria, such as distance-based, information-based, and statistically-based. Filter methods may be univariate or multivariate, determining whether they are capable of picking up feature interactions. Some examples of filter methods are Pearson Correlation and Fast Correlation-Based Filter [9], which filter based on target correlations; Mutual Information [9], Minimum Redundancy Maximum Relevance (mRMR) [10], and Symmetrical Uncertainty [11] which measure the information of features with the target and with each other; and ReliefF [12] and Fisher Score [13] selection which use distance measures between instances to rank features. Filter methods are typically scalable and efficient, as they rely on simple calculations to score the features [6]. However, they are unable to leverage the patterns found during modeling and may result in less performant feature subsets than other methods [14].

Wrapper methods consider the effectiveness of feature subsets by employing a modeling algorithm as a black box evaluator on each subset until finding an optimal subset [6]. Each subset is tested using the chosen model and scored using an appropriate metric, such as, F1 score, and the subset with the best performance is used to train the final model. Wrapper methods require a search strategy in order to effectively explore the feature subsets, which can range from as simple as greedy forward selection or backward elimination to as complex as genetic algorithms and particle swarm optimization [6], [15], [16]. While wrapper methods do tend to select subsets with higher performance than filter based methods [14], they have much higher computational complexity due to the need to iteratively fit models to the data [9]. It is typically only feasible to use simple algorithms as the black box model, such as Naive Bayes or Linear SVM, and there is no guarantee that the subsets found by these models will generalize to the final model used [6].

Embedded methods perform feature selection during or after model training. These methods are either built in to the model itself or are added extensions to the model algorithms. Some common algorithms include CART and C4.5 for decision trees, which control the complexity of the trees by pruning isolated splits or imposing constraints on maximum tree depth [6]. Other common implementations, such as Lasso or ElasticNet, impose regularization penalties to prevent large coefficients and to force small coefficients towards zero, performing feature selection by removing the effect of irrelevant features. A benefit of embedded methods is that they are more efficient than wrapper methods since they do not require iterative retraining of the model, and unlike filter methods, they are able to utilize patterns found by the model itself. However, most embedded methods are model-specific and thus cannot be used across all model types.

B. SHAP Explainability

Shapley Additive Explanations (SHAP) is considered a state-of-the art method for model-agnostic interpretation of black box models. SHAP values are an additive model explanation approach, based on Shapley values from cooperative game theory [7], [17]. SHAP values are typically calculated locally at each data point. Each feature is assigned an importance value that measures how much that feature contributes to the conditional expectation that influences the local data point away from the global expected prediction. For classification tasks, the SHAP values represent the influence of each feature in pushing the classification of a data point towards each class compared to the average class prediction of all data points. The sum of all of the SHAP values equals the original classification by the model. SHAP values satisfy many desirable properties, including local accuracy, missingness, and consistency [7].

While SHAP values are often used to explain individual predictions, they can be used to approximate the global black box model [18]. By taking the absolute value of the SHAP values for each data point and averaging them, the global feature importance for the model is attained [17]. Each of these mean absolute SHAP values provides a global importance of how much the feature influences predictions away from the mean prediction for the entire dataset. This usage of SHAP is used for model explainability, but conveniently this process produces a feature ranking. By sorting these mean absolute SHAP values in decreasing magnitude, the features are scored in a method similar to filter methods while using patterns learned from the model itself, akin to an embedded method. We use this fact as the basis for our feature selection technique. Since SHAP provides a method of interpreting the important features for predictions, it is reasonable to assume that the most important features, as ranked by SHAP, would make for the strongest feature subset.

One important problem to note is that the calculation of SHAP values is known to be an NP-hard problem [19], as sampling all of the possible conditional probabilities for the

Shapley coalitions requires iterating over the same feature subspace of size 2^n discussed earlier. There are several SHAP implementations that seek to solve this problem, including KernelSHAP, PermSHAP, and TreeSHAP. It is beyond the scope of this paper to cover each of these algorithms, but the main consideration is the trade-off between speed, model-specificity, and ability to capture feature interactions. KernelSHAP is a model-agnostic approach that fits a linear regression to the SHAP values, and as such it cannot capture any feature interactions [7], [17]. PermSHAP is also a model-agnostic approach that samples the features in forward and reverse directions using antithetic sampling [19]. This process can be repeated any number of times, but by doing this process exactly once, evaluating the model function $2n + 1$ times per background data sample, up to second-order interactions can be captured [20]. PermSHAP is generally more computationally efficient than KernelSHAP in practice [17]. TreeSHAP utilizes the structure of tree-based algorithms to greatly reduce the complexity of calculating SHAP values. TreeSHAP calculates the exact SHAP values with all feature interactions in $O(TLD^2)$ time, where T is the number of trees, L is the maximum number of leaves in any tree, and D is the maximum depth of any tree [17], [21]. While this is an incredible performance increase over the other discussed implementations, TreeSHAP is specific only to tree-based models.

Due to its model-agnostic nature and ability to capture up to second-order feature interactions, we have elected to use the PermSHAP algorithm for our experiments.

III. Literature Review

The usage of SHAP values for feature selection is a relatively new concept. In [8], the TreeSHAP algorithm was utilized along with XGBoost models across eight small datasets and compared with ANOVA, Mutual Information, and Recursive Feature Elimination feature selection strategies. The strategies employed in [8] are similar to the strategies we propose in this paper; however, by using TreeSHAP, the versatility of SHAP selection for non-tree based models was not tested. In addition, [8] offered no particular strategy for choosing the number of features to keep when using SHAP.

In [22], TreeSHAP was utilized for regression tasks. However, the SHAP selection was combined with a sequential forward selection strategy, requiring many iterations to find the optimal feature subsets, which removes the benefit of using SHAP as a one-pass technique for feature selection.

TreeSHAP and XGBoost models were used in [23] to predict credit card fraud. Rather than use the features as sorted by SHAP values, an additional step of assigning statistical significance to each feature importance was performed. For classification tasks, logistic regression was used on the SHAP values, while for regression tasks, linear regression was used. These feature significance tests were recursively repeated until a final feature set was selected. This paper claims that their algorithm produces better results with fewer features than the naive implementation of SHAP values for feature selection, but the authors only tested a single k value for the number of selected features for the naive implementation, and their algorithm produces only a slight improvement over the naive implementation while requiring three times the runtime. As this paper uses TreeSHAP, the results do not generalize to non-tree based models.

The naive implementation of SHAP values for feature selection was implemented in [24] for predicting credit card fraud. Although several model types were used, such as XGBoost, Decision Trees, CatBoost, Extremely Randomized Trees, and Random Forests, all models were tree-based and utilized TreeSHAP. In addition, the authors utilized a naive selection process, choosing the top 3, 5, 7, 10, or 15 features. Many of their results were inconclusive.

The paper in [25] compared the effectiveness of SHAP selection versus Lasso selection on a breast cancer dataset. Unlike most implementations of SHAP selection, the authors did not use mean absolute SHAP values and instead used raw SHAP values. They selected the top 20 most common features among SHAP explanations for seven different model types and used this feature set for selection, disconnecting the SHAP selection from a specific model type. This paper utilized many model types, including non-tree based models, such as Logistc Regression; however, we were unable to determine which particular SHAP algorithm was used within the paper.

The GitHub repository in [26] proposes the BorutaSHAP algorithm. This repository came to our attention due to being mentioned in passing within [27]; however, it has no associated paper itself. This algorithm is based upon the Boruta features selection algorithm [28] where the feature set is extended with "shadow" copies of each feature. These shadow copies are permuted to remove all correlations with the target, and a model is fit on the extended feature set. Features which perform worse than the shadow features are eliminated along with their shadow counterparts, and this process is performed iteratively until all shadow features have been eliminated. The BorutaSHAP algorithm expands upon the Boruta algorithm by using SHAP values for scoring. The iterative nature of this algorithm is both a blessing and a curse; while it may lead to more confidence regarding the chosen feature subset, the iterative nature leads to a less efficient and more computationally intensive algorithm. In addition, the particular implementation of this model uses TreeSHAP as the underlying SHAP explainer, limiting it to only tree-based models.

The Powershap algorithm proposed in [27] is similar to the BorutaSHAP algorithm already described. However, instead of creating an entire shadow feature set, a single feature is randomly selected to use as a shadow copy during each iteration. This random feature changes for each iteration and after all iterations have been processed, each feature is ranked based on the percentage of iterations where it performed better than the random feature using one-tailed student-t tests. Any feature which is considered significant by this t-test is kept and all others are discarded. The Powershap algorithm is flexible and can be used with more models than just tree-

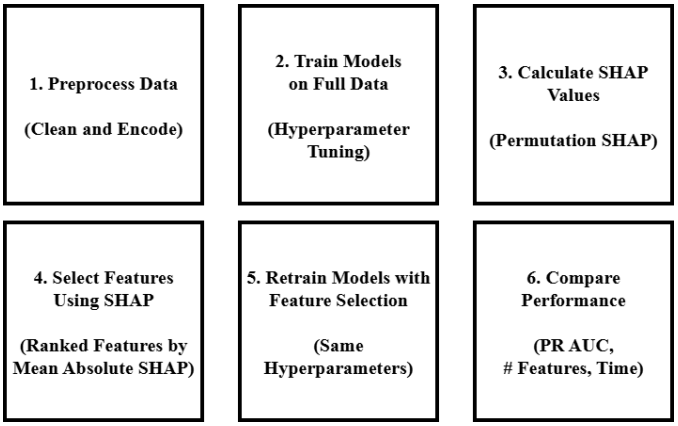

Fig. 1. Overview of the steps for each dataset.

based models; however, it suffers from similar issues as the BorutaSHAP algorithm due to it being an iterative process and being overall more computationally intensive than a one-pass SHAP selection technique.

We used the same one-pass naive algorithm as implemented in [8]. However, unlike many prior experiments, we used a number of machine learning algorithms and the model-agnostic PermSHAP explainer to evaluate the effectiveness and consistency of SHAP selection across different model types without relying on specific model architectures. In addition, rather than only evaluate the results on one dataset, we utilized a diverse set of datasets in order to conduct a more comprehensive evaluation of SHAP selection for datasets of varying complexity. We also have utilized two novel selection strategy implementations in order to try to find optimal feature subsets using a one-pass technique. The details of the datasets used and the feature selection strategies are presented in the following section.

IV. Methodology

A. Overview

The experiment consists of six general steps for each dataset, as outlined in Fig. 1. First, the dataset must be preprocessed in preparation for training. Then, each model is trained and hyperparameter tuned on the full dataset. These "full" models are used to calculate SHAP values with PermSHAP in order to capture feature importances. These SHAP importances are aggregated, and sorted in descending order to create feature rankings. Feature subsets are selected using these feature rankings according to the strategies outlined later in this section. Each model is then retrained on the reduced datasets using the same hyperparameters. Finally, the performance of each model is evaluated on the full and reduced datasets.

This process is repeated for each model and feature selection technique. This means for a particular model-dataset combination, roughly 49 feature sets were extracted and used for training and evaluation; one for the full feature set plus 12 for each of the feature selection techniques: SHAP, Mutual Information Gain (MINFO), ReliefF, and mRMR. As there are five model types tested, this results in about 245 different feature sets tested per dataset and a total of 2,450 experiments across all ten datasets. Not all models produced the full 49 feature sets, as the different SHAP selection strategies outlined later in this section sometimes produced identical feature subsets.

TABLE I
Dataset Information

Dataset Name	Source	Instances	Features
Indian Liver Patient [33]	UCI	584	10
Heart Disease [34]	UCI	303	13
Mushroom [35]	UCI	8,124	22
SPECT Heart [36]	UCI	267	23
Breast Cancer [37]	UCI	569	30
Secondary Mushroom [38]	UCI	61,068	20
Credit Card Fraud [39]	Kaggle	284,807	30
Phishing URL [40]	UCI	235,795	54
Patient Survival [41]	Kaggle	91,700	84
Android Permissions [42]	UCI	29,333	86

Our project utilized the typical Python machine learning libraries Scikit-learn, Pandas and NumPy, along with the shap [29] package for generating SHAP explanations, the ReliefF library [30] for relief-based feature selection, and the Feature-engine library [31] for mRMR selection. All of the files, experiments, and code can be found in our GitHub repository at [32].

B. Datasets Used and Preprocessing

Ten datasets were selected for use in this project. Eight of these datasets come from the UCI Machine Learning Repository, and the remaining two datasets are from Kaggle. These datasets were chosen due to their use in prior papers about feature selection and to test the effectiveness of SHAP across a variety of dataset sizes and complexities. Each dataset represents a binary classification problem. Table 1 outlines the complexity of each dataset in terms of number of instances and number of features prior to preprocessing.

Each dataset was sent through basic preprocessing in preparation for modeling. Duplicate rows and columns with greater than 50% missingness were dropped. Missing values were imputed using the median value for numeric columns and with the mode for categorical columns. Numeric features were standardized, and categorical variables were encoded based upon their cardinality. Categorical features with only two unique values were encoded as binary flags. Categorical features with three to ten unique values were one-hot encoded, creating one column per value of the feature. If a categorical feature contained more than ten unique values, the feature was target encoded, replacing each value with the averaged class prediction of all instances with that value [43]. Not all models handle sparsity well, so this encoding scheme was chosen to prevent extreme increases in data dimensionality. 80% of the data was set aside for training and the remaining 20% was held for testing.

C. Model Training and Tuning

Five models were selected for testing: Logistic Regression (LOGREG), Decision Trees (DT), Random Forest (RF), XGBoost (XGB), and Support Vector Machines (SVC). These models were chosen for their popularity in traditional machine learning and to test the effectiveness of SHAP selection across models with varying degrees of built-in feature selection. Since SHAP values are used to provide explanations for trained model predictions, they are able to leverage the benefits of embedded feature selection that is already present in the modeling algorithms. Thus, the SHAP selection technique we have proposed is able to piggyback on the feature selection already performed internally within each model and is sensitive to hyperparameter tuning.

Each model was trained with an exhaustive search of a small to moderate sized hyperparameter grid. Tuning and scoring was performed using 5-fold cross-validation. Area Under the Precision-Recall Curve (PR AUC) was chosen as the metric to optimize, as a higher PR AUC indicates a more robust model and is more informative, stable, and relevant for datasets with large class imbalances than other common classification metrics, such as Area Under the Receiver Operator Characteristic (ROC AUC) or F1 score [44].

D. Selection Methods and Comparison Metrics

Two different SHAP selection strategies were tested, one focused upon capturing a percentage of total feature importances, and one focused upon capturing strong features relative to the strongest feature. All SHAP explanations were generated using the PermSHAP algorithm. These strategies were labeled SUM and MAX strategies, respectively.

Algorithm 1 SUM SHAP Selection Strategy

Sort the SHAP values p_i in descending order: $p_i \geq p_{i+1}$
Calculate the total sum $S = \sum p_i$
Define a proportion constant r between 0 and 1.
Initialize a running sum s and an index $i = 0$.
while $s < r \cdot S$ **do**
    Add p_i to s and increment i
end while
return The index i for the sorted SHAP values.

The SUM strategy, outlined in pseudo-code in Algorithm 1, works by selecting a number of features such that their combined SHAP values equal some proportion of the sum of all SHAP values. This selection strategy focuses on capturing a meaningful proportion of the feature importances discovered by the model. Higher values of r result in more features chosen. The tested values for r using the SUM strategy were 0.5, 0.6, 0.7, 0.8, 0.9, and 0.95.

The MAX strategy, outlined in pseudo-code in Algorithm 2, compares the relative strength of each feature to the strongest feature, selecting only features that are at least as strong as a user-defined proportion of the strongest feature. This strategy focuses on attempting to capture only strong features, based on the assumption that only the strongest features are important.

Algorithm 2 MAX SHAP Selection Strategy

Sort the SHAP values p_i in descending order: $p_i \geq p_{i+1}$
Choose the largest SHAP value p_0 as the max value M
Define a proportion constant ρ between 0 and 1.
Initialize an index $i = 0$.
while $p_i \geq \rho \cdot M$ **do**
    Increment i
end while
return The index i for the sorted SHAP values.

Lower values of ρ result in more features chosen. The tested values for ρ using the MAX strategy were 0.01, 0.05, 0.1, 0.15, 0.25, and 0.5.

Three filter-based methods were chosen for comparison with SHAP selection: MINFO, ReliefF, and mRMR. Each of these algorithms simply ranks the features and requires the user to select the resulting k features they wish to use for selection. In order to provide useful comparisons, the MAX and SUM SHAP selection strategies were employed to generate candidate k values for each dataset, and these k values were used for each of MINFO, ReliefF, and mRMR.

For each dataset, each model was trained using the full dataset and hyperparameters were selected. The same models with fixed hyperparameters were then retrained on all candidate feature subsets as discovered by the SHAP selection process and all candidate feature subsets as discovered by MINFO, ReliefF, and mRMR. Model performance was scored using 5-fold cross-validation with PR AUC as the primary metric.

The feature selection methods were compared across three categories. The first comparison was how well each feature selection was able to reduce model overfitting between cross-validation scores and test scores, or in ideal cases, to improve generalization scores. This comparison measured the ability of the technique to select relevant features and remove irrelevant or redundant features. The second comparison performed measured the degree of feature reduction each technique was able to achieve. If a feature reduction technique can attain similar or higher scores using fewer features than another technique, then that technique is better at ranking and extracting important features compared to its peers. The third and final comparison performed was a computational efficiency evaluation. While certain techniques may produce better results, if they have are computationally infeasible, their usefulness may be limited in practice.

REFERENCES

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," Journal of Machine Learning Research, vol. 3, pp. 1157–1182, 2003.

[2] H. Abdi and L. J. Williams, "Principal component analysis," WIREs Comput. Stat., vol. 2, no. 4, pp. 433–459, 2010, doi: 10.1002/wics.101

[3] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," Journal of Machine Learning Research, vol. 9, pp. 2579–2605, 2008.

[4] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," arXiv preprint, arXiv:1802.03426, 2018.

[5] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014, 40th-year commemorative issue.

[6] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), May 2015, pp. 1200–1205.

[7] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[8] W. E. Marcílio and D. M. Eler, "From explanations to feature selection: Assessing SHAP values as feature selection mechanism," in Proc. 33rd SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI), Porto de Galinhas, Brazil, 2020, pp. 340–347, doi: 10.1109/SIBGRAPI51738.2020.00053.

[9] Y. B. Wah, N. Ibrahim, H. A. Hamid, S. Abdul-Rahman, and S. Fong, "Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy," Pertanika J. Sci. Technol., vol. 26, no. 1, 2018.

[10] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 8, pp. 1226–1238, Aug. 2005, doi: 10.1109/TPAMI.2005.159.

[11] X. Lin, C. Li, W. Ren, X. Luo, and Y. Qi, "A new feature selection method based on symmetrical uncertainty and interaction gain," Computational Biology and Chemistry, vol. 83, p. 107149, 2019, doi: 10.1016/j.compbiolchem.2019.107149.

[12] R. J. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore, "Relief-based feature selection: Introduction and review," J. Biomed. Inform., vol. 85, pp. 189–203, 2018, doi: 10.1016/j.jbi.2018.07.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046418301400

[13] Q. Gu, Z. Li, and J. Han, "Generalized Fisher score for feature selection," arXiv preprint, arXiv:1202.3725, 2012. [Online]. Available: https://arxiv.org/pdf/1202.3725

[14] B. Xue, M. Zhang, and W. N. Browne, "A comprehensive comparison on evolutionary feature selection approaches to classification," Int. J. Comput. Intell. Appl., vol. 14, no. 2, p. 1550008, 2015.

[15] N. El Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in Proc. 2016 Int. Conf. Eng. MIS (ICEMIS), Sept. 2016, pp. 1–5.

[16] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in Proc. 2014 Sci. Inf. Conf., Aug. 2014, pp. 372–378.

[17] C. Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 3rd ed., 2025. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[18] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods," Entropy, vol. 23, no. 1, p. 18, 2020.

[19] R. Mitchell, J. Cooper, E. Frank, and G. Holmes, "Sampling permutations for Shapley value estimation," J. Mach. Learn. Res., vol. 23, no. 43, pp. 1–46, 2022.

[20] SHAP Documentation – PermutationExplainer API, [Online]. Available: https://shap.readthedocs.io/en/latest/generated/shap.PermutationExplainer.html

[21] S. M. Lundberg, G. G. Erion, and S. I. Lee, "Consistent individualized feature attribution for tree ensembles," arXiv preprint, arXiv:1802.03888, 2018. [Online]. Available: https://arxiv.org/abs/1802.03888

[22] Y. Gebreyesus, D. Dalton, S. Nixon, D. De Chiara, and M. Chinnici, "Machine learning for data center optimizations: Feature selection using Shapley additive explanation (SHAP)," Future Internet, vol. 15, no. 3, p. 88, 2023, doi: 10.3390/fi15030088.

[23] E. Kraev, B. Koseoglu, L. Traverso, and M. Topiwalla, "Shap-Select: Lightweight feature selection using SHAP values and regression," arXiv preprint, arXiv:2410.06815, 2024, doi: 10.48550/arXiv.2410.06815.

[24] H. Wang, Q. Liang, J. T. Hancock, et al., "Feature selection strategies: A comparative analysis of SHAP-value and importance-based methods," J. Big Data, vol. 11, no. 44, 2024, doi: 10.1186/s40537-024-00905-w.

[25] M. S. H. Shaon, T. Karim, M. S. Shakil, and M. Z. Hasan, "A comparative study of machine learning models with LASSO and SHAP feature selection for breast cancer prediction," Healthcare Anal., vol. 6, p. 100353, 2024, doi: 10.1016/j.health.2024.100353.

[26] E. Keany, "BorutaShap: A wrapper feature selection method which combines the Boruta feature selection algorithm with Shapley values," Zenodo, 2020, doi: 10.5281/zenodo.4247610.

[27] J. Verhaeghe, J. Van Der Donckt, F. Ongenae, and S. Van Hoecke, "Powershap: A power-full Shapley feature selection method," in Mach. Learn. Knowl. Discov. Databases (ECML PKDD 2022), M. R. Amini et al., Eds. Cham: Springer, 2023, vol. 13713, pp. 49–65, doi: 10.1007/978-3-031-26387-3_5.

[28] M. B. Kursa, A. Jankowski, and W. R. Rudnicki, "Boruta–a system for feature selection," Fundam. Inform., vol. 101, no. 4, pp. 271–285, 2010.

[29] SHAP Documentation (Main Page), [Online]. Available: https://shap.readthedocs.io/en/latest/

[30] R. S. Olson, "ReliefF," Zenodo, 2016, doi: 10.5281/zenodo.47803.

[31] S. Galli et al., "feature-engine/feature_engine: v1.2.0," Zenodo, 2022, doi: 10.5281/zenodo.5818272.

[32] J. Gottlieb, M. Chunsen, and A. Yoo, "SHAP-Feature-Selection," GitHub Repository, accessed Nov. 2, 2025. [Online]. Available: https://github.com/JoshuaGottlieb/SHAP-Feature-Selection

[33] B. Ramana and N. Venkateswarlu, "ILPD (Indian Liver Patient Dataset)," UCI Mach. Learn. Repository, 2022. [Online]. Available: https://doi.org/10.24432/C5D02C

[34] A. Janosi, W. Steinbrunn, M. Pfisterer, and R. Detrano, "Heart Disease," UCI Mach. Learn. Repository, 1989. [Online]. Available: https://doi.org/10.24432/C52P4X

[35] "Mushroom," UCI Mach. Learn. Repository, 1981. [Online]. Available: https://doi.org/10.24432/C5959T

[36] K. Cios, L. Kurgan, and L. Goodenday, "SPECT Heart," UCI Mach. Learn. Repository, 2001. [Online]. Available: https://doi.org/10.24432/C5P304

[37] W. Wolberg, O. Mangasarian, N. Street, and W. Street, "Breast Cancer Wisconsin (Diagnostic)," UCI Mach. Learn. Repository, 1993. [Online]. Available: https://doi.org/10.24432/C5DW2B

[38] D. Wagner, D. Heider, and G. Hattab, "Secondary Mushroom," UCI Mach. Learn. Repository, 2021. [Online]. Available: https://doi.org/10.24432/C5FP5Q

[39] Credit Card Fraud Detection Dataset, version 1, Zenodo, Dec. 2022, doi: 10.5281/zenodo.7395559. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[40] A. Prasad and S. Chandra, "PhiUSIIL Phishing URL (Website)," UCI Mach. Learn. Repository, 2024. [Online]. Available: https://doi.org/10.1016/j.cose.2023.103545

[41] M. Agarwal, Patient Survival Prediction [Dataset], 2021. [Online]. Available: https://www.kaggle.com/datasets/mitishaagarwal/patient

[42] A. Mathur, "NATICUSdroid (Android Permissions)," UCI Mach. Learn. Repository, 2021. [Online]. Available: https://doi.org/10.24432/C5FS64

[43] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," ACM SIGKDD Explor. Newsl., vol. 3, no. 1, pp. 27–32, Jul. 2001, doi: 10.1145/507533.507538

[44] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," PLOS ONE, vol. 10, no. 3, p. e0118432, 2015, doi: 10.1371/journal.pone.0118432.