An Empirical Comparison of Filter, Wrapper, and Hybrid Approaches for Feature Selection

Orin Cohen
orincoh@post.bgu.ac.il
Ben Gurion University of the Negev

Abstract

Choosing the right features is an important step in building good machine learning models, especially when working with datasets that have many irrelevant or repeated features. Keeping only the features that really help can make models faster to train, easier to understand, and less likely to overfit.

This project compares three main approaches: filter methods, wrapper methods, and hybrid methods. The filter methods used include Mutual Information, Symmetrical Uncertainty, Chi-Square, ReliefF, and ANOVA F-test, which rank features based on how relevant they are. The wrapper methods tested were Recursive Feature Elimination with Cross-Validation, Sequential Forward Selection, and Backward Elimination, which choose features by testing how well they help a Random Forest model perform. The hybrid methods combine filter and wrapper or embedded steps to see if doing both together improves results.

Tests were run on different benchmark datasets of various sizes, types, and class balance. The same train-test split and cross-validation were used for all methods. Hyperparameters were tuned to get the best macro F1 score. The results show that there is no single best method for every case. Sometimes wrappers and hybrids work better but take longer to run. Filters are faster but need careful tuning to decide how many features to keep.

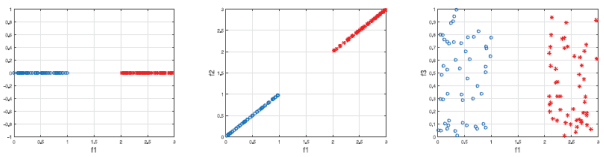
These findings help show what to expect when picking a feature selection method in real projects. Knowing the trade-offs between time, accuracy, and simplicity can help make smarter choices when building machine learning models.

Keywords: feature selection, machine learning, feature ranking, filter approach, wrapper approach, hybrid approach, random forest

1. Introduction

In modern machine learning tasks, datasets often contain many input features, but not all of them contribute equally to predictive performance. Irrelevant or redundant features can add noise, increase computational cost, and reduce model interpretability. Feature selection aims to identify and retain only the informative variables, helping to build simpler and more robust models.

A feature is considered *relevant* if it contributes unique, predictive information, *redundant* if it repeats information already captured by another feature, and *irrelevant* if it adds no useful signal for separating classes. Figure 1 illustrates these concepts: f_1 can clearly distinguish two clusters (classes), f_2 is redundant since it is strongly correlated with f_1, and f_3 adds only noise. Removing redundant or irrelevant features typically improves performance without loss of useful information.



(a) relevant feature f_1 (b) redundant feature f_2 (c) irrelevant feature f_3

Figure 1: Illustration of relevant (f_1), redundant (f_2), and irrelevant (f_3) features (Li et al., 2017).

However, despite decades of research, choosing suitable features for a given task is still not straightforward. The main approaches, which include filter, wrapper, embedded, and hybrid methods, each have their own strengths and limitations. No single method consistently works best for all types of data or learning problems. In practice, balancing dimensionality reduction with good predictive performance remains a practical and often dataset-dependent challenge.

In this project, I compare filter, wrapper, and hybrid feature selection approaches on multiple datasets with diverse characteristics. The selected feature subsets are then evaluated using a Random Forest classifier to examine how different selection strategies affect classification results, dimensionality, and computational cost.

The aim is not to provide a universal solution but to highlight practical differences and trade-offs that can help when applying feature selection to real-world problems.

2. Literature Review

Feature selection is a fundamental task in machine learning, particularly in high-dimensional settings where many features may be redundant or irrelevant. By identifying a subset of informative features, models can achieve better accuracy, reduced training time, and improved interpretability. Over the past decades, a wide range of feature selection methods have been developed and refined, including filter-based, wrapper-based, embedded, and hybrid approaches.

One of the earliest systematic comparisons between filter and wrapper methods was conducted by Kohavi and John (1997), who introduced the wrapper approach. Ac-

cording to this approach, feature subsets are evaluated based on the performance of a predictive model. Their empirical study on eight real-world datasets demonstrated that wrappers tend to outperform filters in terms of accuracy, particularly when feature dependencies exist. However, they also emphasized the computational cost of wrappers, which limits their scalability.

Filter methods, in contrast, evaluate features independently of the learning algorithm using statistical or information-based measures. Common techniques include Information Gain, Chi-square, ReliefF, and Symmetrical Uncertainty (Theng and Bhoyar, 2024). These methods are typically fast and scalable, making them suitable for high-dimensional datasets. However, they may overlook interactions between features since each feature is assessed in isolation.

Wrapper methods use a predictive model to evaluate the quality of feature subsets. The selection process involves searching through subsets of features - often using strategies such as Sequential Forward Selection, Sequential Backward Elimination, Genetic Algorithms, and Beam Search and scoring them based on model accuracy (Theng and Bhoyar, 2024). While this approach often yields better performance than filters, it comes at the cost of significantly higher computational complexity.

Embedded methods select important features during the model training process and are usually tailored to the specific learning algorithm.

Guyon and Elisseeff (2003) provided a comprehensive overview of filter, wrapper, and embedded approaches and their theoretical foundations. Their work highlighted key evaluation criteria such as relevance (how useful a feature is for prediction) and redundancy (how much overlapping information exists among features). They emphasized the importance of considering both aspects when selecting a feature subset. Additionally, the paper discussed the risks of overfitting, particularly when using complex feature sets in high-dimensional spaces. Although not empirical in nature, their review served as a foundational reference and offered practical guidelines for applying feature selection across various domains.

Following these foundational works, researchers began to explore hybrid strategies aimed at balancing the efficiency of filters with the performance of wrappers. Das (2001) proposed a boosting-based hybrid feature selection method, where AdaBoost is used not only to improve classification but also to guide feature subset selection by iteratively adjusting the weights of the examples. The approach, which incorporates characteristics of both filters and wrappers, was evaluated on six real-world datasets from the UCI repository. Results showed that the hybrid method often achieved competitive or superior performance compared to traditional wrappers, while significantly reducing computational cost.

Further empirical studies examined how these methods behave across various datasets and classifiers. Janecek et al. (2008) evaluated multiple filter and wrapper methods and analyzed their effect on classification performance. Their findings showed that wrapper methods, which assess feature subsets in the context of a specific learning algorithm, often outperform filter methods. Additionally, they emphasized that reducing the number of features can lead to better accuracy, but the effectiveness of the method strongly depends on the nature of the dataset. Karabulut et al. (2012) presented an empirical comparison of six filter-based feature selection techniques across 15 datasets and three classifiers. Despite being concise, their study suggests that multilayer perceptrons benefit more from feature selection than decision trees, highlighting the role of the classifier in the effectiveness of different methods.

A more recent trend has been to categorize feature selection methods not just by algorithm type but also by data characteristics. Li et al. (2017) introduced a data-centric taxonomy that takes into account whether the data is static, streaming, structured, or heterogeneous. This perspective highlights the importance of aligning the selection method with the nature of the dataset.

More recently, Theng and Bhoyar (2024) provided an updated perspective, reviewing over two decades of research on feature selection methods. Their work aimed to address the challenges that stem from the large number of existing feature selection algorithms and the complexity of choosing the right method for a given task. They proposed a detailed taxonomy of feature selection methods based on evaluation criteria (e.g., univariate vs. multivariate), learning setting (supervised, semi-supervised, unsupervised), search strategy (sequential, random, exponential), and computational complexity. Unlike previous surveys that focused primarily on either algorithmic aspects or data characteristics, this paper emphasized both perspectives simultaneously. In addition to reviewing classical methods, the authors discussed emerging directions such as deep learning-based selection, fuzzy logic, and evolutionary approaches. They also outlined key open challenges, including model stability, scalability to high-dimensional data, and the need for methods that generalize across domains.

Overall, the literature illustrates the evolution of feature selection from basic filter and wrapper methods to more sophisticated hybrid and data-adaptive techniques. This progression informs the current work, which focuses on empirically comparing selected filter and wrapper methods across datasets, and evaluating whether sequentially combining them can enhance classification performance.

3. Methodology

3.1. Datasets

In this study, six datasets were employed to evaluate and compare feature selection methods. Three datasets were imported directly through the scikit-learn library: Breast Cancer Wisconsin (Diagnostic), Iris, and Wine. These datasets are sourced from the UCI Machine Learning Repository and are available in a preprocessed format through scikit-learn, making them easy to apply.

The remaining three datasets, Mobile Price Classification, Mushrooms, and Zoo, were obtained as CSV files from public repositories, including Kaggle and the UCI repository. These datasets required minimal preprocessing, such as removing identifier columns and encoding categorical target variables into numerical formats.

The datasets were intentionally selected to differ in the number of samples, the number and type of features (numerical, categorical, or mixed), and the number of target classes. This diversity enables a robust comparison of feature selection methods across various data characteristics. The datasets were kept in their original class distributions, which include a mix of balanced, nearly balanced, and imbalanced cases, to allow investigation of how feature selection methods perform under realistic conditions.

The datasets are widely used benchmarks that are relatively clean and required only minimal preprocessing to be ready for analysis. Specifically, all datasets were split into an 80%/20% train-test partition. The identifier column (animal name) in the Zoo dataset was removed. For the Mushroom and Zoo datasets, one-hot encoding was applied to all categorical features, resulting in an increase in the number of features compared to the original datasets. Additional dataset details are summarized in Table 1.

Table 1: Datasets summary table.

Dataset	# of Samples	# of Features	Type	# of Classes
Breast Cancer Wisconsin (Diagnostic)	569	30	Numeric	2
Mobile Price	2,000	20	Mixed	4
Iris	150	4	Numeric	3
Wine	178	13	Numeric	3
Mushroom	8,124	$22 \rightarrow 117$	Categorical	2
Zoo	101	$17 \rightarrow 21$	Categorical	7

The "# of Features" column shows the number of features before and after preprocessing (e.g., column removal and one-hot encoding).

3.2. Feature Selection Methods

This project applied a diverse set of feature selection methods across three main categories: filter, wrapper, and hybrid approaches.

Filter Methods. The filter methods included Mutual Information (MI), Symmetrical Uncertainty (SU), ReliefF, Chi-Square, and ANOVA F-test. These methods independently assess the relevance of each feature using statistical or information-based measures, without relying on any specific predictive model. To determine how many features to retain in the filter-based methods, a selection threshold was applied. Instead of using a fixed top-k approach, features were selected using a threshold determined by the logarithmic heuristic $\log_2(n)$, where n is the number of features. This strategy offers a practical balance by adapting the number of retained features to the dimensionality of each dataset while avoiding arbitrary or overly rigid cutoffs. This choice was guided by previous studies suggesting that there is no universally accepted rule for determining the number of features to retain in filter methods. Common practices such as selecting a fixed top-k, a fixed percentage (e.g., top 10% or 25%), or using an absolute relevance threshold (e.g., SU > 0.05) are widely used but can be problematic. Fixed k or percentile cutoffs may not account for differences in feature space size across datasets, while fixed thresholds may not generalize well across scoring metrics or data types. Another strategy is to look for an "elbow point" in the ranked relevance scores, where a sudden drop might suggest a natural cutoff. However, this method is often subjective, difficult to apply on small datasets, and in some cases the relevance scores decline gradually, making the elbow indistinct. The $\log_2(n)$ heuristic has been previously applied in several studies as a simple, interpretable, and size-adaptive rule (Gao et al., 2011, Balogun et al., 2020, Fernández-Delgado et al., 2014, Hosni et al., 2017, Morán-Fernández and Bolón-Canedo, 2024).

Specifically, MI captures the shared information between a feature and the target variable without assuming linearity. SU normalizes MI to account for bias toward features with more values and provides a symmetrical measure of dependency. ReliefF (Kononenko, 1994) extends the original Relief algorithm (Kira and Rendell, 1992) by supporting multi-class problems and handling noisy and incomplete data. It estimates feature importance based on how well feature values distinguish between neighboring instances of different classes. Chi-Square, applied only to categorical datasets, tests for significant associations between categorical features and the target. ANOVA F-test, used exclusively on continuous datasets, assesses whether the means of a continuous feature differ significantly across target classes.

Wrapper Methods. The wrapper methods included Recursive Feature Elimination with Cross-Validation (RFECV), Sequential Forward Selection (SFS), and Backward Elimination. These methods evaluate subsets of features based on their predictive performance using a particular learning algorithm. In this project, a Random Forest classifier was chosen as the estimator across all wrapper methods because it is also the classifier used in the final training and evaluation stages, ensuring alignment between feature selection and the model used. The selection process was guided by performance optimization using the F1-macro score, which is suitable for multiclass and imbalanced datasets as it equally weights performance across all classes.

In particular, RFECV iteratively removes the least important features while monitoring validation performance through cross-validation to automatically find the optimal number of features; SFS starts with an empty set of features and progressively adds those that improve the model's performance; and Backward Elimination begins with the full set of features and removes them step by step, typically without using internal cross-validation to decide when to stop, retaining only those that contribute meaningfully to predictive accuracy.

Hybrid Methods. Hybrid feature selection strategies aim to combine the strengths of different approaches, typically by leveraging the speed of filter methods alongside the model-awareness of wrapper or embedded techniques. Two hybrid feature selection strategies were explored in this project. The first approach followed a filter-then-wrapper design, where the MI method was first used to filter out irrelevant features based on their relevance scores, and then the RFECV method was applied to the reduced set. This strategy aimed to leverage the computational efficiency of filter methods while refining feature selection through wrapper-based evaluation. The second approach combined a filter method (MI) with an embedded method (Random Forest feature importance) by assigning each feature a weighted score. MI scores were computed and normalized, while feature importances were extracted from a Random Forest classifier, averaged across stratified cross-validation folds and normalized. Both normalized

scores were then linearly combined using fixed weights to produce a final score per feature. Features were selected based on a threshold applied to this combined score. This combination aims to identify features that are both statistically relevant and predictive within the model context.

3.3. Classification Algorithm

Random Forest was selected as the classification algorithm for all experiments due to its robustness, flexibility, and proven performance across a wide range of classification tasks. Importantly, it supports both binary and multiclass classification, which was essential given the diversity of datasets used in this project. In addition, Random Forest is capable of handling heterogeneous data types and imbalanced class distributions, and it provides built-in estimates of feature importance. Taken together, these characteristics made Random Forest a reliable and adaptable choice across the different datasets and evaluation conditions in this project.

3.4. Evaluation

The evaluation of model performance in this project relied primarily on classification metrics appropriate for both binary and multiclass problems, particularly in the presence of class imbalance. The main metric used throughout model tuning and feature selection was the F1 score with macro averaging (F1-macro), which computes the harmonic mean of precision and recall for each class separately, and then averages the results equally across all classes. This ensures that minority classes are treated fairly and prevents dominant classes from skewing the evaluation. Metrics based on macro averaging are commonly recommended for multiclass classification with imbalanced distributions (Grandini et al., 2020).

In addition to F1-macro, macro-averaged precision (precision-macro) and macro-averaged recall (recall-macro) were reported to provide a more detailed view of the model's behavior. All metrics were computed on the test set after model training. Accuracy was not used in this evaluation, given its limitations in handling imbalanced or multiclass data where it may overrepresent majority class performance. To assess the efficiency of each feature selection method, two additional indicators were considered: the number of selected features, and the runtime of the selection process. Performance on the training set was also monitored to identify potential overfitting, although all model selection decisions were based solely on test set results.

3.5. Experimental Setup

All experiments were conducted using Python with standard machine learning libraries. Each dataset was split into training (80%) and test (20%) sets with stratified sampling to maintain the original class distribution. For the Mushroom and Zoo datasets, categorical features were encoded only after the split to prevent information leakage, and the test sets were aligned with the training feature space.

Feature selection was performed exclusively on the training data, with filter methods selecting features based on the $\log_2(n)$ heuristic to adapt the number of retained features to the dataset's dimensionality. For hybrid methods, the initial filter threshold was slightly lower to allow the wrapper or embedded stage to refine the final subset. Wrapper methods used stratified k-fold cross-validation to evaluate feature subsets, with five folds in most cases and three folds for the Zoo dataset due to its small class sizes. The Random Forest classifier was used throughout, both for feature selection and final classification.

Model hyperparameters were tuned with *RandomizedSearchCV* to maximize the F1-macro score, using the following grid: n_estimators {100, 300, 500}, max_depth from 8 to 14, criterion {gini, entropy}, max_features {3, 5, 7}, min_samples_split {2, 5, 10}, and min_samples_leaf {1, 2, 4}. Final performance was evaluated on the hold-out test sets, reporting F1-macro, precision-macro, and recall-macro. A fixed random seed (42) was used throughout to ensure reproducibility.

4. Results

This section presents the performance results for all datasets, focusing on an aggregated view by selection approach (filter, wrapper, and hybrid) rather than describing each method individually. The results are reported as averages of the different methods within each approach (see Table 2). This presentation allows a clearer comparison of how each strategy performed across different datasets in terms of overall classification performance, using the three metrics mentioned earlier (F1-macro, recall-macro, and precision-macro).

Among the **categorical datasets**, the Zoo and Mushrooms sets show a consistent pattern. Both demonstrate that wrapper and hybrid approaches reached perfect test-set results (averages of 1.0 for all metrics) while the filter methods produced lower values (for example, the Mushrooms filter approach around 0.97 and the Zoo filter between 0.74 and 0.81). This difference shows that for clean and separable categorical data, wrapper-based methods can fully exploit informative feature subsets, whereas filter criteria alone may retain redundant or less relevant variables.

In the Wine dataset, which includes **only continuous features**, nearly all approaches produced perfect or near-perfect values across all metrics. The wrapper and hybrid averages were consistently 1.0, with the filter approach slightly lower, around 0.99. A similar trend appears in the Breast Cancer dataset, where all approaches performed very well across all scores, but the wrapper approach reached slightly higher averages (around 0.95) than the filter (around 0.93) or hybrid (around 0.94).

The Iris dataset showed no difference at all: the filter, wrapper and hybrid approaches all produced identical average results (around 0.97 across all metrics). This is reasonable because the dataset contains only four numerical features that already provide clear separation between the three classes. In fact, some selection methods even kept fewer than four features, yet still reached the same scores, showing that the dataset's structure leaves little room for further optimization.

In contrast, the Mobile dataset highlights a different pattern. Here, the filter and wrapper approaches achieved very

similar results, both with averages around 0.93 to 0.94, but the hybrid approach dropped sharply to about 0.75. This drop may be due to the **mixed nature of the data**, which combines continuous and binary attributes. When a hybrid method removes features in stages, it may accidentally discard informative variables too early, or fail to adjust the selection when variables of different types interact, which can limit the final performance.

Detailed scores for each method and dataset are provided in Appendix A.

Table 2: Average test set performance by approach for each dataset. All metrics (F1, Recall, and Precision) are macro-averaged.

Dataset	Approach	F1	Recall	Precision
Breast Cancer	Filter	0.933	0.938	0.928
	Wrapper	0.949	0.946	0.953
	Hybrid	0.943	0.943	0.943
Iris	Filter	0.967	0.967	0.970
	Wrapper	0.967	0.967	0.970
	Hybrid	0.967	0.967	0.970
Mobile	Filter	0.938	0.938	0.939
	Wrapper	0.932	0.932	0.933
	Hybrid	0.752	0.750	0.758
Wine	Filter	0.993	0.994	0.992
	Wrapper	1.000	1.000	1.000
	Hybrid	1.000	1.000	1.000
Zoo	Filter	0.766	0.81	0.743
	Wrapper	1.000	1.000	1.000
	Hybrid	1.000	1.000	1.000
Mushrooms	Filter	0.968	0.967	0.971
	Wrapper	1.000	1.000	1.000
	Hybrid	1.000	1.000	1.000

Regarding the **efficiency measures** of the feature selection stage itself, two indicators were evaluated: the runtime of the feature selection process and the number of features selected by each method.

Table 3 presents the **runtime** by dataset and approach, showing clear differences between filter and wrapper strategies. As can be seen, the runtime values for the filter-based methods were, in most cases, less than one second or just a few seconds. The higher maximum value for filters is mainly caused by the ReliefF method, which generally takes longer because it repeatedly samples instances and compares them with their nearest neighbors. This process is more computationally intensive compared to the simpler statistical tests used in classical filter methods, such as ANOVA F and Chi-Square.

In contrast, wrapper-based methods required significantly longer runtime values, especially for larger datasets such as Mushrooms, where the feature selection process took several hours to complete. This is due to the iterative nature of wrapper methods, which repeatedly train and evaluate models on different feature subsets to find the optimal combination. A detailed table presenting the runtime values for each method is provided in Appendix B.

Regarding the **number of features kept**, some variation was observed across the approaches. Filters like MI, SU and ReliefF typically ranked and limited features, while Chi-Square and ANOVA F-test relied on p-values and tended to keep more. Wrapper and hybrid approaches aimed to balance subset size with predictive performance, and therefore in many cases retained a larger number of features. For simpler datasets such as Iris, most methods across all approaches reduced the original four features to just two. A detailed table presenting the selected feature counts can be found in Appendix C.

Table 3: Runtime values (seconds) by dataset and feature selection approach.

Dataset	Approach	Min	Max	Range
Breast Cancer	Filter	0.01	9.62	9.61
	Wrapper	42.1	369.54	327.44
	Hybrid	1.26	27.45	26.19
Iris	Filter	0.01	0.74	0.73
	Wrapper	4.56	5.7	1.14
	Hybrid	0.76	3.63	2.87
Mobile	Filter	0.21	111.22	111.21
	Wrapper	58.85	285.14	226.29
	Hybrid	0.2	2.3	2.1
Wine	Filter	0.01	2.6	2.59
	Wrapper	15.49	48.86	33.37
	Hybrid	0.84	11.85	11.01
Mushrooms	Filter	0.04	648.63	648.59
	Wrapper	265.01	8,671.75	8,406.74
	Hybrid	6.07	36.7	30.63
Zoo	Filter	0.01	2.87	2.86
	Wrapper	13.14	124.23	111.09
	Hybrid	0.86	11.22	10.36

As a further validation step, the importance scores from the trained Random Forest models were reviewed. In all cases, the features that were selected also showed higher importance in the final model, which helps support the validity of the selection results.

5. Discussion

This project offered an empirical view of how filter, wrapper and hybrid feature selection strategies behave across diverse datasets with varying types, sizes and class distributions. The results show that no single approach can be declared universally superior; however, clear patterns did emerge. Wrapper and hybrid approaches often achieved slightly higher performance scores, particularly on well-separated or mostly categorical data, but required much longer runtime values due to their iterative search. Filter methods, by contrast, were consistently faster and easier to apply but their effectiveness depended heavily on the choice of how many top-ranked features to keep.

These findings illustrate a practical trade-off: achieving slight improvements in classification performance with wrappers or hybrids comes at the cost of significant computation, which may limit their feasibility for larger or more complex problems. For simple or low-dimensional data such as the Iris dataset, all approaches performed similarly, suggesting that when the signal is already clear, extensive selection adds little practical benefit.

To ensure consistent comparisons, only the Random Forest classifier was used. This choice ensured all pipelines could be evaluated under the same conditions, but it also introduced a limitation: Because Random Forest gives feature importance scores by itself, this may have partly covered for the weaknesses of simpler filter methods. Therefore, using a model that ranks features automatically can blur the true differences between the selection strategies.

When designing the experiment, a diverse mix of

datasets was included, covering small clean numeric examples and larger categorical or mixed-type sets. However, due to practical computational limits, very large or truly high-dimensional datasets were not included, leaving open questions about how these methods scale under more demanding conditions.

A further point highlighted here is the practical challenge of deciding how many features to retain when using ranking-based filter methods. This study used a simple logarithmic rule to adapt the cutoff to the dataset's dimensionality. While easy to apply, this heuristic cannot guarantee an optimal balance every time. This trade-off is well recognized in the literature: selecting too few features may lead to underfitting, while including too many can introduce noise and hurt generalization, underscoring a practical challenge when using ranking-based filters.

6. Future Work

Future studies could expand on these results in several ways. One direction is to test the same selection pipelines on larger and more complex real-world datasets, particularly those with greater redundancy and noise, to better understand the balance between runtime and performance in practice. Another useful extension would be to repeat the same experiments using classifiers that do not embed feature importance information, such as SVM or k-NN, to better isolate how each selection strategy performs when the model itself does not compensate for poor feature choices. Finally, further research into adaptive or data-driven ways to set thresholds for ranking-based filter methods could make them more reliable competitors to more computationally demanding wrappers and hybrids.

7. References

- Balogun, A. O., Basri, S., Mahamad, S., Abdulkadir, S. J., Almomani, M. A., Adeyemo, V. E., ... & Bajeh, A. O. (2020). Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study. *Symmetry*, 12(7), 1147. https://doi.org/10.3390/sym12071147

- Das, S. (2001, June). Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the 18th International Conference on Machine Learning* (Vol. 1, pp. 74–81).

- Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1), 3133–3181.

- Gao, K., Khoshgoftaar, T. M., Wang, H., & Seliya, N. (2011). Choosing software metrics for defect prediction: An investigation on feature selection techniques. *Software: Practice and Experience*, 41(5), 579–606. https://doi.org/10.1002/spe.1043

- Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: An overview. *arXiv preprint arXiv:2008.05756*. https://doi.org/10.48550/arXiv.2008.05756

- Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar), 1157–1182.

- Hosni, M., Idri, A., & Abran, A. (2017, October). Investigating heterogeneous ensembles with filter feature selection for software effort estimation. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement* (pp. 207–220). https://doi.org/10.1145/3143434.3143456

- Janecek, A., Gansterer, W., Demel, M., & Ecker, G. (2008, September). On the relationship between feature selection and classification accuracy. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery* (pp. 90–105). PMLR.

- Karabulut, E. M., Özel, S. A., & Ibrikci, T. (2012). A comparative study on the effect of feature selection on classification accuracy. *Procedia Technology*, 1, 323–327. https://doi.org/10.1016/j.protcy.2012.02.068

- Kira, K., & Rendell, L. A. (1992, July). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 129–134).

- Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2), 273–324.

- Kononenko, I. (1994, April). Estimating attributes: Analysis and extensions of RELIEF. In *European Conference on Machine Learning* (pp. 171–182). Berlin, Heidelberg: Springer.

- Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6), 1–45. https://doi.org/10.1145/3136625

- Morán-Fernández, L., & Bolón-Canedo, V. (2024). Finding a needle in a haystack: Insights on feature selection for classification tasks. *Journal of Intelligent Information Systems*, 62(2), 459–483. https://doi.org/10.1007/s10844-023-00823-y

- Theng, D., & Bhoyar, K. K. (2024). Feature selection techniques for machine learning: A survey of more than two decades of research. *Knowledge and Information Systems*, 66(3), 1575–1637. https://doi.org/10.1007/s10115-023-02010-5

Appendices

Appendix A: Detailed Performance Metrics

Table A.1: F1-Macro scores for each feature selection method across all datasets.

Approach	Method	Breast Cancer	Iris	Mobile	Wine	Mushrooms	Zoo
Filter	SU	0.926	0.967	0.938	1.000	0.923	0.810
	MI	0.935	0.967	0.932	1.000	0.976	0.810
	ReliefF	0.935	0.967	0.945	0.972	0.972	0.679
	ANOVA F-test	0.934	0.967	0.940	1.000	1.000	1.000
	Chi-Square	–	–	–	–	1.000	1.000
Wrapper	RFECV	0.962	0.967	0.940	1.000	1.000	1.000
	SFS	0.943	0.967	0.927	1.000	1.000	1.000
	Backward Elimination	0.943	0.967	0.927	1.000	1.000	1.000
Hybrid	Hybrid (filter + wrapper)	0.943	0.967	0.927	1.000	1.000	1.000
	Hybrid Weighted (filter + embedded)	0.943	0.967	0.927	1.000	1.000	1.000

Table A.2: Recall-Macro for each feature selection method across all datasets.

Approach	Method	Breast Cancer	Iris	Mobile	Wine	Mushrooms	Zoo
Filter	SU	0.935	0.967	0.938	1.000	0.921	0.857
	MI	0.941	0.967	0.932	1.000	0.976	0.857
	ReliefF	0.941	0.967	0.945	0.976	0.972	0.714
	ANOVA F-test	0.937	0.967	0.940	1.000	1.000	1.000
	Chi-Square	–	–	–	–	1.000	1.000
Wrapper	RFECV	0.957	0.967	0.940	1.000	1.000	1.000
	SFS	0.943	0.967	0.927	1.000	1.000	1.000
	Backward Elimination	0.938	0.967	0.927	1.000	1.000	1.000
Hybrid	Hybrid (filter + wrapper)	0.943	0.967	0.927	1.000	1.000	1.000
	Hybrid Weighted (filter + embedded)	0.943	0.967	0.927	1.000	1.000	1.000

Table A.3: Precision-Macro for each feature selection method across all datasets.

Approach	Method	Breast Cancer	Iris	Mobile	Wine	Mushrooms	Zoo
Filter	SU	0.920	0.970	0.939	1.000	0.936	0.786
	MI	0.930	0.970	0.933	1.000	0.976	0.786
	ReliefF	0.930	0.970	0.945	0.970	0.972	0.657
	ANOVA F-test	0.932	0.970	0.940	1.000	1.000	1.000
	Chi-Square	–	–	–	–	1.000	1.000
Wrapper	RFECV	0.967	0.970	0.940	1.000	1.000	1.000
	SFS	0.943	0.970	0.927	1.000	1.000	1.000
	Backward Elimination	0.948	0.970	0.927	1.000	1.000	1.000
Hybrid	Hybrid (filter + wrapper)	0.943	0.970	0.927	1.000	1.000	1.000
	Hybrid Weighted (filter + embedded)	0.943	0.970	0.927	1.000	1.000	1.000

Appendix B: Selection Runtime Values

Table B.1: Runtime values (seconds) for each feature selection method across all datasets.

Approach	Method	Breast Cancer	Iris	Mobile	Wine	Zoo	Mushrooms
Filter	SU	0.15	0.03	0.21	0.07	0.88	4.54
	MI	0.14	0.02	0.25	0.05	0.33	4.42
	ReliefF	9.62	0.74	111.22	2.60	2.87	648.63
	ANOVA F-test	0.02	0.03	0.24	0.05	0.02	0.06
	Chi-Square	0.01	0.01	0.21	0.01	0.01	0.04
Wrapper	SFS	369.54	5.00	233.96	48.86	120.19	6643.96
	RFECV	42.10	4.56	58.85	15.49	13.14	265.01
	Backward Elimination	360.50	5.70	285.14	48.84	124.23	8671.75
Hybrid	Hybrid (filter + wrapper)	27.45	3.63	0.20	11.85	11.22	36.70
	Hybrid Weighted (filter + embedded)	1.26	0.76	2.30	0.84	0.86	6.07

Appendix C: Selected Feature Counts

Table C.1: Selected feature counts for each feature selection method across all datasets.

Approach	Method	Breast Cancer	Iris	Mobile	Wine	Zoo	Mushrooms
Filter	SU	5	2	5	4	5	7
	MI	5	2	5	4	5	7
	ReliefF	5	2	5	4	5	7
	Chi-Square	5	2	5	4	16	109
	ANOVA F-test	25	4	4	13	16	13
Wrapper	RFECV	22	2	5	6	19	18
	SFS	15	2	10	6	10	58
	Backward Elimination	15	2	10	7	11	59
Hybrid	Hybrid (filter + wrapper)	17	2	1	6	13	19
	Hybrid Weighted (filter + embedded)	22	3	1	11	16	21