Universitat Autònoma de Barcelona
Bachelor's degree in Artificial Intelligence
Advisor: Ernest Valveny Llobet

Benchmark of feature selection techniques for supervised machine learning on tabular data

Miguel Moral Hernández
Miguel.Moral@autonoma.cat

February 25, 2025

Abstract

This research provides a comprehensive benchmark study of feature selection techniques for supervised machine learning on tabular data. The study evaluates several feature selection methods across synthetic and real-world datasets with diverse characteristics. Unlike other studies that focus on specific domains or techniques, this research provides an extensive evaluation framework that assists data scientists to intuitively select appropriate feature selection methods based on their objectives, constraints and specific data characteristics. The comparison of the methods employs multiple performance metrics, with supervised evaluation possible for synthetic datasets, and unsupervised evaluation for real-world datasets. By running experiments in controlled (synthetic) and uncontrolled (real-world) environments, this research offers practical insights into the effectiveness of the different feature selection techniques across different scenarios.

Keywords: feature selection, supervised learning, machine learning, tabular data

1 Introduction

Feature selection (Kumar & Minz, 2014) is considered one of the most frequent and important preprocessing steps in tabular machine learning pipelines, playing an essential role in building efficient and accurate predictive models. The main objective of feature selection is to identify and retain the most relevant subset of features while eliminating redundant or non-informative ones. This process offers multiple advantages that directly impact model performance and practicality. First, reducing the number of input features decreases model complexity and improves computational efficiency by lowering training and inference time. Moreover, it enhances model interpretability, a key aspect in domains where understanding the model outputs is as important as the predictions themselves. Second, it helps to mitigate the curse of dimensionality (Crespo Márquez, 2022), a phenomenon where model performance decreases as the number of features grows disproportionately in comparison to the number of samples. This issue is even more relevant in today's world, where the exponential growth in data volume and complexity has led to datasets with an overwhelming number of features. Therefore, effective feature selection becomes crucial, as it enables data scientists to search through the immense amount of available information and identify the most relevant data. Third, it eliminates noisy or irrelevant features, which often results in improved model accuracy and generalization capability, as it reduces the risk of overfitting (Ying, 2019) and helps the model focus only on the relevant patterns of the data. Additionally, feature selection can help to reduce data storage costs in practical applications, as fewer features need to be measured and maintained.

In today's data-driven landscape, these benefits of feature selection become particularly critical for tabular data (Bobbit, 2022), which has emerged as the cornerstone of machine learning applications across multiple domains. Unlike other types of data such as images or text, where deep learning models can automatically learn relevant features (LeCun, Bengio, & Hinton, 2015), tabular data requires a more detailed approach for selecting and handling features. This is especially true for tree-based models (Analytics Vidhya, 2024a), which, despite their remarkable predictive power on tabular data (Grinsztajn, Oyallon, & Varoquaux, 2022) (Uddin & Lu, 2024), are particularly sensitive to irrelevant or redundant features. These models iteratively divide the dataset by selecting features that optimize specific criteria like gini impurity or information gain at each split. When irrelevant features are present, they can lead to suboptimal splits, increasing model complexity and the risk of overfitting. Through effective feature selection, these issues can be mitigated, ensuring that tree-based

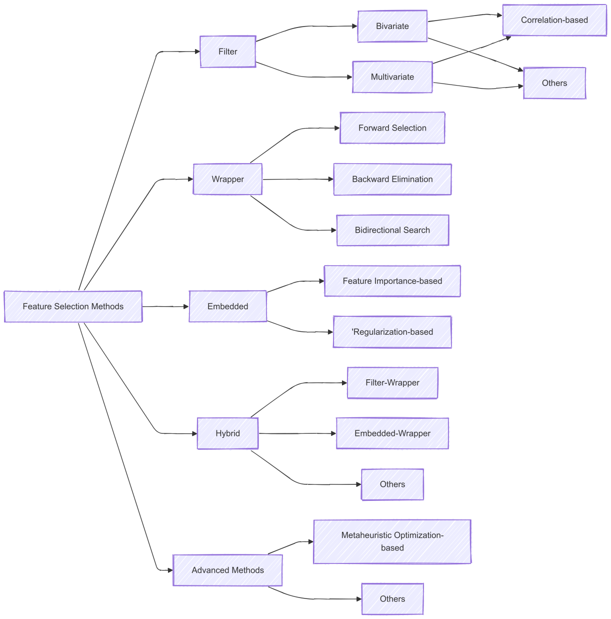

Figure 1: Categorization of feature selection techniques

models maintain their efficiency and predictive strength by operating on a cleaner and more relevant feature set.

Despite the clear benefits of feature selection, there not exists a universally superior technique for selecting the best features across all scenarios. The effectiveness of each method can have a high variance depending on the characteristics of the dataset, the predictive task and the model used. This creates a significant problem for data scientists, who must face the difficult task of selecting the right technique without clear guidance on which will be the most effective in their specific problem. This challenge forms the main motivation and objective of the study. By testing a wide variety of feature selection techniques across different scenarios, the study aims to offer professionals a clear guide that will help them to intuitively and efficiently decide which is the best technique for their specific case and objectives. Our comprehensive analysis shows the advantages and disadvantages of each technique in different situations, offering a robust framework for making informed decisions about feature selection strategies.

Therefore, the main contributions of this study are as follows:

- A comprehensive comparative analysis of various feature selection techniques evaluated on multiple and diverse data scenarios, examining the trade-offs between different approaches in both controlled (synthetic) and uncontrolled (real-world) environments (Figures **3** and **4**) . Through detailed evaluation across multiple performance metrics, this analysis provides data scientists with clear insights into the strengths and limitations of each method, enabling informed decisions based on their specific requirements and constraints.
- Development of a sophisticated synthetic data generation pipeline that creates controlled environments with four distinct feature types (informative, interactive, redundant and noise features), incorporating complex mathematical relationships and providing importance scores for relevant features to enable a precise evaluation of feature selection methods.
- Development of a dual evaluation framework that employs different metrics depending on whether datasets are real-world or synthetic. The framework applies unsupervised metrics when evaluating real-world datasets and supervised metrics for synthetic datasets. Additionally, execution time is measured for both types of data sources.

2 State of the art revision

Before diving into the details of our research, it is essential to understand the current landscape of feature selection techniques and related research. This section first provides a comprehensive overview of feature selection methods across five major categories (Section **2.1**). Moreover, the literature review then identifies research gaps in feature selection evaluation, particularly focusing on the limited scope of previous benchmarking studies and the need for a more comprehensive analysis across diverse scenarios (Section **2.2**).

2.1 Feature selection techniques

Figure **1** defines the categories and some subcategories that are considered in this study. The figure provides a comprehensive overview of different types of feature selection techniques and their hierarchical relationships. Let's dive into the 5 families of feature selection methods:

1. **Filter methods**: These approaches (Fritz, 2023b) select features based on their intrinsic properties, independent of any specific learning algorithm. Filter methods operate through two main approaches. **Bivariate filter methods** evaluate features individually by examining their relationship with the target variable using criteria such as mutual information or correlation coefficients, ranking and selecting features based on these measures. **Multivariate filter methods** evaluate the feature space as a whole. These methods consider relationships among multiple features, assessing their combined impact on the target variable. By evaluating features in groups rather than in pairs, multivariate methods can capture dependencies and interactions among features that influence the target variable. Examples include Minimum Redundancy

Maximum Relevance (mRMR), Relief approaches, Fast Correlation-Based Filter (FCBF) and others.

2. **Wrapper methods**: These methods (Medium, 2021a) use machine learning algorithms to evaluate feature subsets, iteratively selecting features that optimize model performance metrics such as accuracy or AUC. Unlike filter methods, they take a learning algorithm into account, which can result in more relevant selected features for the model but higher computational cost. These systematic methods follow fixed recursive procedures through the feature space using four main strategies. **Exhaustive selection** evaluates all possible feature combinations, **forward selection** incrementally adds features starting from none, **backward elimination** iteratively removes features starting with all features and **bidirectional search** combines forward and backward approaches simultaneously to find a unique solution.

3. **Embedded methods**: These techniques (Fritz, 2023a) perform feature selection within the model construction process itself, optimizing feature selection and model training simultaneously. This integration leads to greater efficiency compared to wrapper methods, which require recursive feature evaluation based on model scores. Embedded methods operate through two main mechanisms. **Importance-based approaches** rank features based on their contribution to the model's performance. Features with higher importance scores are more likely to be retained in the final model, while less important ones may be eliminated. Examples include tree-based feature importance approaches and permutation feature importance. **Regularization-based approaches** add penalties (L1 or L2) to the model's objective function, shrinking irrelevant feature coefficients toward zero.

4. **Advanced methods**: These refer to newer and more sophisticated techniques that extend traditional feature selection approaches including metaheuristic optimization-based approaches, Boruta, SHAP, LIME and others. **Metaheuristic optimization-based methods** are probabilistic algorithms that use various strategies to explore the feature space algorithm. As well as wrapper methods, metaheuristic optimization-based approaches use learning algorithms to recursively calculate model scores. Examples include genetic algorithms, simulated annealing or particle swarm optimization.

5. **Hybrid methods**: These approaches (Medium, 2020) combine different feature selection techniques to enhance both efficiency and accuracy. While any combination of feature selection methods can theoretically be integrated to create a hybrid approach, certain combinations have proven particularly effective in practice. **Filter-Wrapper** methods reduce computational cost by first using a filter method to eliminate irrelevant features before applying wrapper techniques to the reduced feature set. **Filter-Embedded** methods similarly begin with a filter approach before employing embedded methods. **Embedded-Wrapper** methods first rank features using importance measures, then apply wrapper methods to identify optimal feature subsets, with Recursive Feature Elimination (RFE) serving as a famous example.

2.2 Benchmarks of feature selection techniques

Even though numerous studies have explored and compared feature selection techniques, there exists a significant research gap in the extensive evaluation of these methods across different data contexts. Previous benchmarking studies have typically focused on specific aspects of feature selection. For instance, some studies have exclusively evaluated filter methods (Sánchez-Maroño, Alonso-Betanzos, & Tombilla-Sanromán, 2007), while others have analyzed techniques within particular domains such as disease risk prediction (Pudjihartono, Fadason, Kempa-Liehr, & O'Sullivan, 2022) or multi-omics data (Li, Mansmann, Du, & Hornung, 2022). These narrowly focused studies are valuable for specific applications but leave professionals without a clear idea when facing different types of data and challenges. Consequently, this research addresses the identified research gap by offering an extensive comparison of feature selection methods across all major categories (Figure **1**) and diverse data conditions (Figure **2**), providing data scientists with practical guidelines for selecting appropriate feature selection techniques based on their specific context and requirements.

3 Data

To properly understand the project, is important to introduce the concepts of synthetic and real-world dataset. A synthetic dataset constists of artificially generated data where there is a complete knowledge of how features are generated and related to target variable. This means there is a precise knowledge of which features contribute to the target (known as relevant features) and which ones do not (known as irrelevant features). In contrast, real-world datasets are collected from real scenarios, and although general relationships in the data can be understood, there is typically no perfect knowledge of which features truly influence the target variable. Even more challenging, there is no guarantee that the features of the dataset contain sufficient predictive power or valuable information to effectively model our target. This key difference makes synthetic datasets particularly valuable for evaluating feature selection methods, as the selected features can be compared against the known relevant ones. With real-world

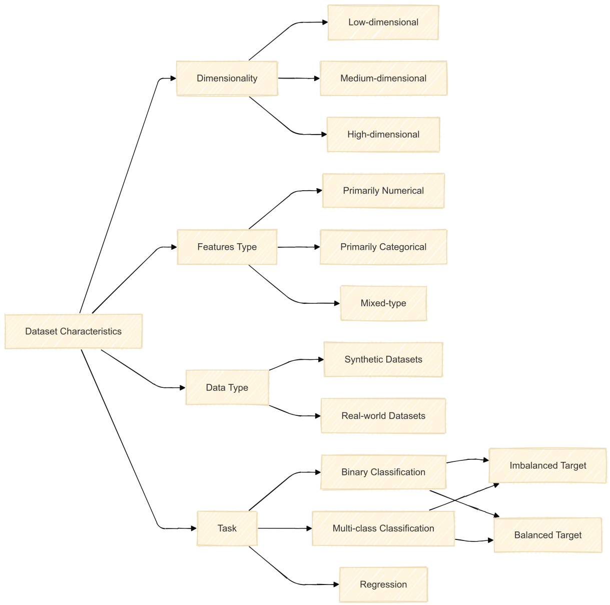

Figure 2: Data Characteristics. The end nodes of the figure are non-excluding nodes with respect to the terminal nodes of different branches. A dataset can be high-dimensional, with primarily numerical features, regression and synthetic.

datasets, this evaluation is more challenging and requires unsupervised techniques since ground truth about feature relevance is not available.

Comprehensively evaluating feature selection techniques required using 100 datasets, comprising 50 synthetic and 50 real-world datasets. As illustrated in Figure **2**, datasets are categorized across multiple domains to ensure comprehensive evaluation. Regarding dimensionality, datasets are classified as high-dimensional when they contain more than 100 features, mid-dimensional when they have between 11 and 100 features, and low-dimensional when they have 10 or less features. In terms of variable types, datasets are categorized as primarily numerical or primarily categorical when at least 80% of their features belong to one of these types, with the remaining cases classified as mixed-type. For the type of task, datasets are categorized into regression problems when the target is numerical and continuous and classification problems otherwise. Within classification tasks, they are distinguished between binary and multiclass problems, with specific criteria for class imbalance. Binary classification datasets are considered imbalanced when one class represents more than 80% of the samples, while in multiclass scenarios, the threshold for imbalance is set at 60%.

The datasets were designed and retrieved to create a balanced representation across these characteristics, spanning different dimensionalities, task types, feature compositions and target distributions. However, some dataset characteristics appear less represented due to practical and availability constraints. In Appendix **8.3** you can find a detailed explanation of these constraints and the exact characteristic distributions of the 100 datasets. Despite these limitations, the research prioritized maintaining the distribution of data characteristics as balanced as possible while ensuring the quality and reliability of the selected datasets. A detailed description of each dataset is available in Table **2** of Appendix.

3.1 Synthetic data

To establish a controlled testing environment, a comprehensive framework was developed for generating synthetic datasets with known relevant features and precise mathematical relationships. 50 synthetic datasets were generated, with sample sizes spanning from 1,000 to 50,000 and containing between 4 and 130 features. Throughout the generation process, purely numerical features were created to establish clear and traceable relationships between features and target variables. The subsequent sections will detail the intricate process of generating these synthetic datasets.

3.1.1 Base data generation

The initial feature values are generated sampling from diverse probability distributions including normal, lognormal, exponential, beta, gamma, weibull and chi-square distributions. This variety ensures that the base features exhibit different statistical properties and relationships, imitating the complexity often found in real-world data. To maintain consistent scale and comparability between features, the distribution parameters are carefully controlled, with most values falling within similar numerical ranges. You can find the used distributions with the corresponding parameters in Table **7** of Appendix.

3.1.2 Feature types

Each dataset contains four distinct types of features, carefully designed to represent different aspects of real-world data while maintaining clear mathematical relationships:

1. **Informative features**: These features directly contribute to the target variable through linear or nonlinear relationships (quadratic, sinusoidal, exponential, etc.).

   Each informative feature is generated as:

   $$X_{informative} = \beta \cdot f(X) \tag{1}$$

   where:

   - β is positive or negative coefficients randomly sampled from clipped normal distribution [1.0, 5.0].
   - f is either:

  - Identity function with probability $1 - p$
  - One of {square, sin, exp, log, sqrt, tanh} with probability p

  For example, if X_1 is an informative feature, it might contribute to the target as $2.5{*}X_1$ (linear) or $3.1{*}(X_1^2)$ (nonlinear).

2. **Interactive features**: These features don't work in isolation, but rather interact with other features, representing complex relationships often found in real-world scenarios. For instance, in baking, oven temperature alone doesn't predict cake texture (since any temperature could be good or bad), but when combined with cooking time, it becomes critical; as high temperatures require shorter times and low temperatures need longer times. Interactions between features are implemented through either multiplication or division operations, where an interactive feature can form relationships with either informative features or other interactive features.

   Each interaction is generated as:

$$X_{interaction} = \gamma \cdot f(X_i \otimes X_j) \tag{2}$$

   where:

   - X_i is the interactive feature
   - X_j is either an informative or another interactive feature
   - γ is positive or negative coefficients randomly sampled from clipped normal distribution [1.0, 5.0].
   - $\otimes \in \{*, /\}$ is either multiplication or division
   - f is applied to either X_i or X_j (not both) and is either:
     - Identity function with probability $1 - p$
     - One of {square, sin, exp, log, sqrt, tanh} with probability p

   For example, if X_1 is interactive and X_2 interactive/informative, their target contribution could be $1.8 * (sqrt(X_1) * X_2)$ or $2.3 * (X_1/X_2)$.

3. **Redundant features**: These features are generated from informative features through controlled linear combinations and added noise. Each redundant feature is derived from an informative feature, capturing only a portion of its information and incorporating some noise. This combination results in a feature that exhibits correlation with the target but provides less reliable and weaker explanatory power than its parent informative feature. Therefore, they are irrelevant and often difficult to detect as non-informative.

   Each redundant feature is generated as:

$$X_{redundant} = \alpha X_{informative} + \epsilon \tag{3}$$

   where:

   - α is positive or negative coefficients randomly sampled from clipped normal distribution [1.0, 5.0].
   - ϵ is Gaussian noise with scale proportional to the standard deviation of $X_{informative}$ (20-80% of std).

   For example, if X_1 is an informative feature, a redundant feature might be generated as $X_{redundant} = 2.1 * X_1 + N(0, 0.4 * std(X_1))$.

4. **Noise features**: The remaining features maintain their original statistical distributions without any modifications or relationships to the target variable. These features serve as distractors for feature selection methods, simulating irrelevant variables often present in real-world datasets.

3.1.3 Target generation

The target variable is generated through a precise mathematical formula combining the relevant features (informative and interactive):

$$y = \sum_{i=1}^{n} \beta_i f_i(X_i) + \sum_{j=1}^{m} \gamma_j h_j(X_j, X_{p(j)}) \tag{4}$$

where:

- n represents the number of informative features and m the number of interactive features
- X_j represents an interactive feature
- $X_{p(j)}$ represents the partner feature for interactive Xj, which can be either:
  - An informative feature (X_i where $i \leq n$)
  - Another interactive feature (X_k where $k \leq m$ and $k \neq j$)
- $h_j(X_j, X_{p(j)})$ represents one of these two interaction forms:
  - $f_j(X_j) \otimes X_{p(j)}$
  - $X_j \otimes f_j(X_{p(j)})$
- $\otimes \in *, /$ represents either multiplication or division
- f_i and f_j represent possible nonlinear transformations {square, sin, exp, log, sqrt, tanh} with probability p
- β_i, γ_j are positive or negative coefficients randomly sampled from clipped normal distributions [1.0, 5.0]
- No additional noise is added to maintain perfectly controlled environment

For classification tasks, this continuous target is discretized using carefully controlled thresholds to maintain the desired class balance ratios.

3.1.4 Feature importance calculation

To enable precise evaluation of feature selection methods in synthetic scenarios, this research presents an approach for calculating feature importance scores. The goal is to assign each feature in the target formula (informative and interactive features) a normalized importance score representing its relative contribution to the target variable. The importance calculation algorithm follows these steps:

1. **Calculate term-wise effects**:

   A term in the target formula represents either a single feature with its coefficient and possible nonlinear transformation (e.g., $2.5X_1$ or $3.0\log(X_1)$), or an interaction between features with its coefficient and a possible nonlinear transformation applied to one of the features (e.g., $1.8(X_2 * X_3)$ or $4.2(\log(X_2) * X_3)$). The algorithm first breaks down the target formula into these individual terms. For each term, the effect is calculated by evaluating the term result for each data sample, taking the absolute value, and averaging across all samples. This process quantifies how much each term contributes to the target variable's value.

2. **Assign feature contributions**:

   When a feature appears in a single-feature term, the entire effect calculated for that term is assigned to that feature. For interaction terms where two features are involved, the term's effect is divided equally among the participating features. This division reflects that interacting features share responsibility for their joint impact on the target variable.

3. **Normalize importance scores**:

   The raw importance score for each feature is calculated by summing all of its contributions from the terms where it appears, whether from single-feature terms or its share of interaction terms. These raw scores are then normalized to percentages by dividing each feature's total contribution by the sum of all term effects and multiplying by 100, ensuring that feature importance scores sum to 100%.

4. **Compile relevant features**:

   The features are arranged in descending order based on their calculated normalized importance scores. This final list includes both informative features and interactive features, providing a complete ranking of feature relevance to the target variable.

To illustrate this process, let's consider an example target formula:

$$y = 2.5X_1 + 1.8(X_2 * X_3) + 3.0\log(X_4)$$

The formula contains three terms, a single-feature linear term $2.5X_1$, an interaction linear term $1.8(X_2 * X_3)$, and a single-feature nonlinear term $3.0\log(X_4)$. First, the mean absolute effect is computed for each term across all data samples. The effects of single-feature terms are assigned directly, so the effect of $|2.5X_1|$ is assigned to X_1 and the effect of $|3.0\log(X_4)|$ is assigned to X_4. For the interaction term, the effect $|1.8(X_2 * X_3)|$ is splited equally between features X_2 and X_3. Finally, each feature's importance score is computed by summing its assigned raw feature effects and normalizing to percentages.

This detailed importance calculation enables more sophisticated evaluation of feature selection methods in synthetic datasets, recognizing that features contribute differently to the target variable's generation.

3.2 Real-world data

To complement the controlled environment of synthetic datasets, this study incorporates 50 real-world datasets sourced from established repositories, including Kaggle, UCI Machine Learning Repository and OpenML. The datasets belong to diverse domains where data scientists commonly face feature selection challenges, such as financial analytics, educational assessment, healthcare diagnostics and others. This collection of real-world datasets provides a practical counterpart to the synthetic datasets, enabling evaluation of feature selection methods in scenarios where relevant features that contribute to the target are unknown, and data relationships may be more complex than in controlled environments.

4 Methodology

This study compares 20 feature selection techniques, described in Table **1** of Appendix, tested within 100 datasets (50 real-world and 50 synthetic) shown in Table **2** of Appendix. This section details complete methodological framework of the project. The discussion begins by describing the specific feature selection methods considered in the study and important implementation restrictions (Section **4.1**). The experimental setup is then presented, focusing on cross-validation implementation, threshold-based feature selection approaches and model selection strategy (Section **4.2**). The following section then details the performance metrics used both for method execution and evaluation (Section **4.3**), and concludes with the presentation of a dual evaluation approach for synthetic and real-world datasets (Section **4.4**).

4.1 Feature selection techniques

This study evaluates 20 feature selection techniques, described in Table **1** of Appendix, that represent key approaches across the methodological categories outlined in Section **2.1**. The implemented methods are available in public Python repository **feature-selection-benchmark** (Moral, 2025). However, due to both computational constraints and method-specific applicability requirements, certain feature selection techniques were restricted to specific datasets. For detailed information about these restrictions, see Appendix **8.5**. Even though many feature selection techniques are available, evaluating all of them is not possible due to practical limitations. Therefore, several important types of methods have been excluded from this research for the following reasons. First, **exhaustive search feature selection (deterministic-wrapper method**), cannot be included due to its computational complexity of $O(2^n - 1)$ with n representing features. While this approach would theoretically return the optimal feature subset by trying all possible combinations, it becomes computationally infeasible even with moderately sized datasets. For example, a dataset with just 20 features would require evaluating over one million combinations. Sequential wrapper methods such as Sequential Backward Selection (SBS) address this limitation by finding near-optimal solutions with reduced computational cost, making them applicable to higher dimensional datasets. Similarly, **metaheuristic optimization-based methods (advanced)** such as genetic algorithms and particle swarm optimization are excluded due to their substantial computational requirements. Although these methods are designed to find near-optimal solutions without exploring the entire search space, their iterative nature and population-based approach still demand significant computational resources, often requiring many generations to converge to a satisfactory solution. Additionally, **regularization-based feature selection techniques (embedded)** such as lasso and elastic net are excluded because they are specifically designed for linear models by penalizing coefficients. While feature engineering techniques could potentially help these methods capture nonlinear relationships and complex interactions between features, implementing such transformations would be prohibitively time-consuming and falls outside the scope of this study. For synthetic datasets, these methods would be inadequate as the data is intentionally generated with nonlinear relationships and feature interactions, as explained in Section **3.1**. The study also makes the reasonable assumption that real-world datasets contain significant nonlinear relationships between features and target variables. Therefore, methods designed specifically for linear models would not effectively identify these relationships without extensive analysis and feature engineering.

4.2 Experimental setup

The experimental setup of this study is designed to maximize both the reliability and practical applicability of feature selection methods. Three fundamental design choices shape the implementation strategy, focusing on cross-validation integration, threshold-based feature selection and model selection.

4.2.1 Cross-validation implementation

Cross-validation (Berrar et al., 2019) is integrated into all methods requiring model performance calculations. This includes wrapper methods and any technique that evaluates feature subsets with model metrics such as model scores, shapley values, importance scores and others. The use of validation data, separated from training data, is essential to obtain unbiased estimates of model performance. Cross-validation extends this principle by dividing the data into multiple folds, providing multiple validation sets rather than relying on a single train-test split. This approach prevents overfitting by providing more robust performance estimates, thus ensuring that selected features maintain their predictive power on unseen data. For detailed information about the cross-validation strategies used in this study, see Appendix **8.6**.

4.2.2 Threshold-based feature selection

Although not explicitly mentioned in all method descriptions of Appendix Table **1**, techniques operate based on thresholds rather than a predetermined number of features to select. This approach differs from many existing studies that evaluate methods by fixing the number (or ratio) of features to be selected, which can be problematic since the number of relevant features in a dataset is not known in advance. The effectiveness of these methods heavily depends on their thresholds. Some methods such as chi-squared and information value have well-established standard thresholds from statistical theory and practice, while others require more careful consideration. To facilitate intuitive threshold selection, many of the implemented methods normalize their scores (importance scores) to the range [0,1], making it easier to set meaningful thresholds across different scenarios. Users can adjust these thresholds according to their needs, with higher thresholds being more restrictive and selecting fewer features, while lower thresholds are more permissive and select more features. The complete list of thresholds used for each feature selection method, along with their interpretations, can be found in Table **8** of Appendix.

4.2.3 Model selection

Predictive models are required for two distinct purposes in this study. First, they are needed for the execution of certain feature selection methods (wrapper, embedded, hybrid and advanced) shown in Table **1** of Appendix. Second, they provide a means of unsupervised evaluation in real-world datasets by generating model scores (e.g., accuracy) on selected features. For both purposes, tree-based models are used rather than deep learning approaches. This decision is based on recent research findings demonstrating that tree-based models consistently outperform neural networks when working with tabular data (Grinsztajn et al., 2022).

Their superior performance in tabular data scenarios makes them ideally suited for both feature selection and evaluation tasks.

For feature selection methods requiring a predictive model, CatBoost is used as the base algorithm (with the exception of Random Forest importance method, which naturally uses Random Forest). This choice is supported by multiple studies that have demonstrated CatBoost's superior performance compared to other tree-based models (Ye, Liu, Cai, Zhou, & Zhan, 2024) (Dorogush, Ershov, & Gulin, 2018) (Shmuel, Glickman, & Lazebnik, 2024). For unsupervised evaluation with real-world datasets, CatBoost is also used to generate model scores.

To maintain focus on the primary objective of evaluating feature selection methods, hyperparameter optimization (T. Yu & Zhu, 2020) has been excluded from this study. This decision was made due to the significant computational costs involved, particularly for wrapper methods using recursive feature selection, where optimization would be required for each new feature subset of features. While this approach may result in lower absolute model scores, it prevents overfitting and maintains computational feasibility. Therefore, basic and fast hyperparameters are used for both CatBoost and Random Forest throughout the study. You can find the used hyperparameters in Appendix **8.8**

4.3 Model performance metrics

Performance metrics are needed for two essential purposes in this study. First, they act as scoring functions for feature selection methods that require model scores, such as wrapper methods or permutation importance embedded method. Second, they enable unsupervised evaluation of real-world datasets by measuring the performance of the model on selected features.

4.3.1 Mean Absolute Error

For regression tasks, Mean Absolute Error (MAE) (Medium, 2024) is used:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5}$$

- y_i is the true value for the data point i
- $\hat{y}_i$ is the predicted value for the data point i
- n is the total number of data points

MAE is chosen over other metrics like Root Mean Squared Error (RMSE) (Jim, 2023) because it's more robust to outliers and provides more interpretable results, as the error is in the same units as the target variable.

4.3.2 ROC AUC

For classification tasks (binary and multiclass) with balanced target distributions, ROC AUC (Area Under the Receiver Operating Characteristic Curve) (Evidently AI Team, 2024) is used. ROC AUC score evaluates how well a model can distinguish between classes across different classification thresholds. It plots True Positive Rate (TPR) against False Positive Rate (FPR):

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN} \tag{6}$$

A perfect classifier achieves a TPR of 1.0 and FPR of 0, indicating that it correctly identifies all positive cases without any error. ROC AUC was chosen over other metrics like F1-score or accuracy because it provides a threshold-independent evaluation of model performance, making it especially valuable for comparing different feature selection methods where optimal decision boundaries may vary. For multiclass cases, it is calculated using a one-vs-rest approach, where each class is evaluated against all others combined.

4.3.3 Average Precision

For imbalanced classification tasks (binary and multiclass), Average Precision (AP) score is used instead. AP score summarizes the precision-recall curve by calculating the average of precision values, weighted by the change in recall at each threshold:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{7}$$

$$AP = \sum_{n}(R_n - R_{n-1})P_n \tag{8}$$

where:

- R_n is recall at threshold n
- P_n is precision at threshold n
- n represents the index of the threshold at which precision and recall are calculated.

Unlike ROC AUC, which can be highly optimistic in imbalanced datasets due to the large number of true negatives affecting its FPR, Average Precision (AP) provides a more reliable evaluation by excluding true negatives from its calculation. AP uses precision and recall measurements, making it more sensitive to false positives in imbalanced scenarios. For instance, in a dataset with 990 negative and 10 positive cases, if a model predicts 100 samples as positive (capturing all 10 true positives but also 90 false positives), ROC AUC would get good performance (100% TPR, 9%

FPR) due to the high number of true negatives, while AP would clearly reveal the poor performance through its low precision (10/100 = 10%). In critical domains like fraud detection or disease diagnosis, where positive cases are rare, AP effectively measures how well the model ranks positive instances above negative ones. This makes AP particularly valuable for imbalanced problems, where both missing a positive case (like a disease or fraud) and generating excessive false alarms can have significant consequences. For multiclass cases, the target labels are first transformed into a binary matrix using one-vs-rest encoding, and then the AP score is computed on this binarized representation.

4.4 Feature selection evaluation metrics

The evaluation process uses different metrics depending on dataset type. The framework applies unsupervised metrics when evaluating real-world datasets and supervised metrics for synthetic datasets. However, execution time (minutes) is measured for all techniques across all datasets, with lower values being preferred as they indicate better computational efficiency.

4.4.1 Real-world datasets evaluation metrics

Unlike synthetic datasets where relevant features are known, real-world datasets lack information about which features truly influence the target variable. This necessitates an unsupervised evaluation approach based on model performance scores, as features enabling higher predictive power can be reasonably assumed to be more relevant to the target variable. Following this principle, the Percentage Score Error metric is employed. This metric quantifies the performance difference between a model trained on the selected features and the best model performance achieved among the feature selection methods applied to the dataset. Given a real-world dataset, the data is first split into training (80%) and test (20%) sets. Feature selection methods included in Table **1** of Appendix are applied only to the training data, resulting in a subset of selected features. A CatBoost model is then trained on the training set using only the selected features and evaluated on the test set. The specific CatBoost score metric used (e.g., ROC AUC, Average Precision, MAE) varies based on the task type and target distribution, as explained in Section **4.3**.

To calculate the **Percentage Score Error (PSE)**, the following formula is used:

$$PSE = \frac{|Bs - Ms|}{|Bs|} \times 100 \tag{9}$$

where:

- Bs is the highest CatBoost score achieved by any feature selection method tested on the dataset. In regression tasks, the best score is the minimum, while in classification tasks is the maximum.
- Ms is the CatBoost score obtained by the specific feature selection method being evaluated.

The Percentage Score Error (PSE) represents the relative difference between the method's score and the best achieved score, expressed as a percentage. A lower PSE indicates that the feature selection method is able to achieve performance closer to the optimal feature selection technique. However, it is crucial to consider the PSE in conjunction with the number of selected features. A method that achieves a slightly higher PSE while selecting significantly fewer features may be preferable to one that achieves a marginally lower error using many more features. The primary goal of feature selection is to maintain similar predictive power with a reduced set of features, improving model interpretability and reducing computational complexity. Therefore, when evaluating feature selection methods on real-world datasets, the trade-off between the PSE and the number of selected features is carefully considered. The ideal feature selection method should strike a balance between minimizing the PSE and selecting a compact set of relevant features.

The preprocessing applied to real-world datasets before this evaluation is available in Appendix **8.9**

4.4.2 Synthetic datasets evaluation metrics

For synthetic datasets, supervised evaluation is possible due to complete knowledge of which features are truly relevant to the target variable. Unlike real-world datasets where evaluation relies on model performance, synthetic datasets allow direct comparison between selected features and known relevant ones. The synthetic data generation pipeline provides quantified importance scores for each relevant feature, ranging from 0 to 100, representing their contribution to the target variable. This enables a more sophisticated evaluation that considers not just whether a feature was correctly selected, but also its relative importance in relation to the target variable. For example, selecting a feature with 30% of importance is weighted more heavily than selecting one that contributes only 5%.

For synthetic datasets, supervised evaluation uses three key metrics:

1. Relevant Accuracy (RA):

$$RA = \sum_{f \in S \cap R} \frac{I_f}{I_{total}} \tag{10}$$

where:

- S is the set of selected features by the method
- R is the set of relevant features
- I_f is the importance score of feature f
- I_{total} is the sum of all importance scores (100)

2. Irrelevant Accuracy (IA):

$$IA = \frac{|I - S|}{|I|} \tag{11}$$

   where:

   - I is the set of irrelevant features
   - S is the set of selected features by the method
   - $|I - S|$ represents correctly unselected irrelevant features

3. **Weighted Accuracy (WA)**:

$$WA = w_R \cdot RA + w_I \cdot IA \tag{12}$$

   where:

   - $w_R = \frac{|R|}{|F|}$ is the weight for relevant features
   - $w_I = \frac{|I|}{|F|}$ is the weight for irrelevant features
   - $|F|$ is the total number of features
   - RA is the Relevant Accuracy
   - IA is the Irrelevant Accuracy

Weighted Accuracy score is the principal performance metric used to evaluate feature selection methods in synthetic datasets. This metric combines both aspects of feature selection, first correctly identifying relevant features (weighted by their importance) and second correctly excluding irrelevant ones.

5 Results

Our evaluation framework adopts distinct metrics for synthetic and real-world datasets, as detailed in Section **4.4**. The results represent averages of these metrics across applicable datasets for each method (applicability restrictions detailed in Appendix **8.5**). While comprehensive numerical results are available in Tables **13** and **14** of Appendix, the findings are presented through intuitive visualizations that better illustrate the relationships between different performance measures.

For real-world datasets (Figure **3**), a three-dimensional visualization plot is developed to capture the interplay between three critical metrics measured across the 50 real-world datasets:

- Average number of selected features (Y-axis): Represents the average number of features chosen by each method, providing insight into their selectivity.
- Average Percentage Score Error (X-axis): Quantifies the relative performance of each feature selection method by calculating the mean Percentage Score Error (explained in Section **4.4.1**) across all datasets where the method was evaluated. A score closer to zero indicates superior performance, with lower values meaning minimal deviation from the best-performing method's results.
- Average execution time (point size): Represented in minutes by the size of each data point, where larger points indicate longer average execution times.

For synthetic datasets (Figure **4**), since Weighted Accuracy (WA) metric inherently accounts for feature selection efficiency, a simpler two-dimensional visualization plot is developed. Weighted Accuracy metric already penalizes the selection of unnecessary features, making a separate axis for the average number of selected features redundant. The plot presents:

- Average Weighted Accuracy Ranking (X-axis): Represents the overall performance of each method, incorporating both the accuracy of selecting relevant features and the ability to exclude irrelevant ones, as detailed in Section **4.4.2**. Methods achieving highest Weighted Accuracy scores are ranked first (get lower ranking values). If multiple methods achieve the same Weighted Accuracy score, they receive the same ranking value. The ranking ranges from 1 to 20 (corresponding to the number of evaluated techniques).
- Average execution time (Y-axis): Measures the computational efficiency of each method in minutes, presented on a logarithmic scale to better visualize the substantial differences between faster and slower methods.

The analysis of these plots reveals an inherent multi-objective optimization challenge in feature selection, where no single method achieves optimal performance across all metrics. For real-world datasets, the theoretical optimal point would lie at the origin (0,0) of Figure **3** with minimal point size, representing a method with the lowest Average Percentage Score Error, fastest execution time and fewest selected features. Similarly, in synthetic scenarios, an optimal method would position at the bottom-left of Figure **4**, combining minimal Average Weighted Accuracy Ranking with fastest execution time. The absence of such dominating methods creates a Pareto frontier where improvements in one metric often come at the cost of others. While methods closer to the axes' origin (with smaller point sizes for real-world data) demonstrate strong overall performance, declaring universal method superiority becomes impossible. The selection of an appropriate technique depends entirely on application priorities between computational efficiency, selection accuracy and feature set minimization. This understanding of performance trade-offs guides practitioners in choosing methods aligned with their particular constraints and objectives.

6 Conclusions

The obtained results of feature selection techniques across both synthetic and real-world datasets reveals several important insights about their relative effectiveness and practical applicability. The results demonstrate clear trade-offs between computational efficiency, selectivity and performance across different methods. Let's analyze the results (**3**, **4**) by examining each method family.

Filter methods show significant limitations, with most achieving poor performance in both synthetic and real-world scenarios. While Chi-squared achieves a relatively low Average Percentage Score Error (4.38%) in real-world datasets, this comes at the cost of poor selectivity, retaining nearly 57 features on average. Other filter methods like Mutual Information and Information Value follow a similar pattern, moderate error scores but very high feature retention (>50 features). FCBF stands out for its excellent selectivity (5.52 features) but suffers from high error rates (21.88%). The synthetic results further confirm these limitations, with most filter methods achieving poor Average Weighted Accuracy Rankings (11-16 range).

Embedded methods emerge as particularly effective, especially CatBoost Feature Importance and Permutation Feature Importance. CatBoost Feature Importance achieves satisfactory synthetic ranking (5.04) and real-world error (5.14%) while maintaining strong selectivity (12.14 features) and remarkably fast execution times (0.0028/0.083 minutes). Permutation Feature Importance shows similar strength in synthetic ranking (4.82) and real-world metrics (6.35% error, 10.90 features) but requires significantly more computation time (0.249/2.27 minutes). Random Forest Feature Importance, while computationally efficient, demonstrates weaker performance compared to its embedded counterparts, especially in synthetic scenarios.

Wrapper methods demonstrate outstanding effectiveness despite their computational cost. Sequential backward approaches (SBS and SBFS) achieve the lowest Average Percentage Score Error (4.31%) among all methods while maintaining reasonable selectivity (around 13 features). Forward approaches (SFS and SFFS) show slightly higher error rates (6.73-7.06%) but with superior selectivity (around 6.5 features). This pattern is reinforced in synthetic datasets where forward methods achieve exceptional rankings (3.73-3.88) compared to backward approaches (10.29-10.44). The forward and backward floating variants offer only marginal performance improvements while significantly increasing computation time. Even though wrapper methods are highly effective, their high execution times (8-19 minutes) require careful consideration of computational resources.

Among advanced methods, SHAP demonstrates remarkable versatility. It achieves strong synthetic ranking (6.4) and excellent real-world error (4.70%) while maintaining reasonable selectivity (13.68 features) and low execution time (0.329/0.456 minutes). Boruta shows decent performance but requires significantly more computation time while retaining more features.

Hybrid methods demonstrate strong results despite their primary drawback of high computational cost. These methods are specifically designed to scale more efficiently than wrapper approaches to higher dimensional scenarios while maintaining comparable or superior performance. SHAP - SFS emerges as the standout hybrid approach, achieving exceptional performance in both synthetic (2.68 ranking) and real-world scenarios (6.21% error) while maintaining strong selectivity (7.64 features), though requiring significant computation time (10.267/17.196 minutes). RFE shows decent metrics across both contexts (5.80 ranking, 5.14% error, 9.26 features) with better computational efficiency (4.193/16.759 minutes). While MI - SFS shows good performance, its high computational cost (13.701/43.397 minutes) stems from poor selectivity in its initial mutual information phase, creating bottlenecks in the subsequent SFS phase. FCBF - SFS excels in synthetic scenarios (2.7 ranking) but struggles with real-world datasets (18.07% error), reflecting the limitations of its initial FCBF filtering step which shows similar weaknesses in isolation.

Among all evaluated techniques, when computational time is not a constraint, wrapper methods Sequential Forward Selection (SFS) and Sequential Forward Floating Selection (SFFS), along with the hybrid method SHAP - SFS, demonstrate superior results in synthetic rankings and real-world error rates while maintaining strong feature selectivity. The hybrid approaches, particularly SHAP - SFS, demonstrate notable advantages over wrapper methods due to their superior scalability to higher dimensional datasets. This scalability is particularly significant since wrapper methods were restricted to low and mid-dimensional scenarios (Appendix **8.5**), while hybrid methods maintained comparable performance and execution times across all dimensionalities. Therefore, **SHAP - SFS** stands out as the optimal choice when computational resources permit, as it not only scales more efficiently than wrapper methods but often achieves superior performance with excellent selectivity. On the other side, for applications where computational efficiency is crucial, three methods demonstrate excellent balanced results. SHAP, CatBoost Feature Importance and Permutation Feature Importance achieve strong performance in both synthetic and real-world scenarios while maintaining low execution times. Among these computationally efficient approaches, **CatBoost Feature Importance** emerges as the standout method by delivering exceptional results with the lowest computational cost, making it the optimal choice for applications that require both computational efficiency and strong selection accuracy. The study also definitively shows that while filter methods are traditionally chosen for their computational efficiency, their consistently poor results across both scenarios, either through inadequate performance or poor selectivity, makes them unsuitable even as components in hybrid approaches. This is particularly relevant given that CatBoost Feature Importance achieves similar computational efficiency while delivering substantially superior performance, effectively eliminating any practical justification for using filter methods. These findings provide clear guidance for practitioners in selecting appropriate feature selection methods based on their specific requirements regarding computational resources and selection accuracy needs.

6.1 Dimensionality and task results analysis

Following the general result analysis of feature selection methods, a focused examination of results patterns across specific dataset characteristics reveals additional insights. The following analysis explores optimal approaches based on dataset dimensionality and task type, as these represent the most influential data charcteristics factors in feature selection. Complete numerical results for dimensionality and task are available in Tables **15**, **16**, **17** and **18** of Appendix, with detailed analysis in Appendix **8.11**. The key findings from this characteristic-based analysis are summarized below.

The dimensionality analysis reveals distinct optimal approaches for different scale scenarios. For high-dimensional datasets, researchers should prioritize SHAP method, which balance performance with computational efficiency. SHAP - SFS and Recursive Feature Elimination (RFE) hybrid methods, emerge as powerful alternatives, delivering superior performance despite significant computational demands. These approaches excel when peak selection accuracy is critical and computational resources are available. In mid-dimensional datasets, CatBoost Feature Importance, provides an optimal compromise between accuracy, feature selectivity and computational efficiency, with SFS and SFFS wrapper approaches serving as attractive alternatives when computational constraints are flexible. For low-dimensional datasets, embedded methods particularly Permutation and CatBoost Feature Importance demonstrate remarkable effectiveness across all metrics. In these scenarios, SFS and SFFS wrapper methods and hybrid SHAP - SFS become increasingly viable, offering precise feature selection capabilities with manageable computational times.

The task-based analysis highlights optimal feature selection approaches for different machine learning challenges. For regression problems, SHAP - SFS and RFE hybid methods, stand out for their exceptional performance in both synthetic and real-world scenarios. Additionally, SHAP method provides a more computationally efficient alternative while maintaining strong overall selection accuracy. In binary classification, CatBoost Feature Importance emerges as the most balanced choice, offering consistent effectiveness across both contexts with remarkable efficiency, while SHAP and SHAP - SFS present decent alternatives with different computational trade-offs. Multiclass classification is best handled by CatBoost Feature Importance, which achieves outstanding accuracy and selectivity in both scenarios with impressive computational efficiency. Furthermore, SFS and SFFS wrapper approaches and hybrid methods SHAP - SFS and RFE offer superior performance when computation time isn't a limiting factor. For all tasks, if computational efficiency is a concern, embedded methods (particularly CatBoost Feature Importance) consistently provide the best performance-efficiency balance, while hybrid approaches (particularly SHAP - SFS) deliver superior selection accuracy when computational resources permit longer execution times.

7 Future Work

While this study provides comprehensive insights into feature selection techniques, several areas warrant further investigation. Future work should explore the impact of optimized hyperparameters for both the tree-based models used in evaluation and within feature selection methods, which could potentially improve selection accuracy and model performance, though at significant computational cost. Additionally, the performance of many feature selection methods depends heavily on their threshold parameter, and a systematic study of optimal threshold selection across different dataset characteristics could enhance method effectiveness. Another valuable direction would be generating more complex synthetic datasets with broader sampling ranges beyond the current limits (1,000-50,000 samples, 4-130 features), incorporating categorical features and including more sophisticated feature interactions and relationships. This expanded synthetic data generation could better simulate extreme scenarios and edge cases found in real-world applications. Moreover, further investigation would also be valuable in understanding the underlying reasons for certain result patterns observed in the results. For instance, analyzing why Sequential Backward Selection (SBS) and Sequential Backward Floating Selection (SBFS) show excellent results in real-world error rates but significantly poorer rankings in synthetic scenarios could provide important insights into the methods' behavior under different conditions. The strong results demonstrated by both SHAP - SFS and CatBoost Feature Importance also suggest exploring a potential new hybrid method that combines CatBoost Feature Importance with SFS, which could leverage the strengths of both approaches. Finally, removing the dimensional restrictions on wrapper methods and testing more datasets would provide a more complete comparison. While this would require significant computational resources, it could reveal important insights about method performance at scale. These improvements would provide even more robust guidance for practitioners while potentially revealing new insights about feature selection effectiveness across different circumstances.

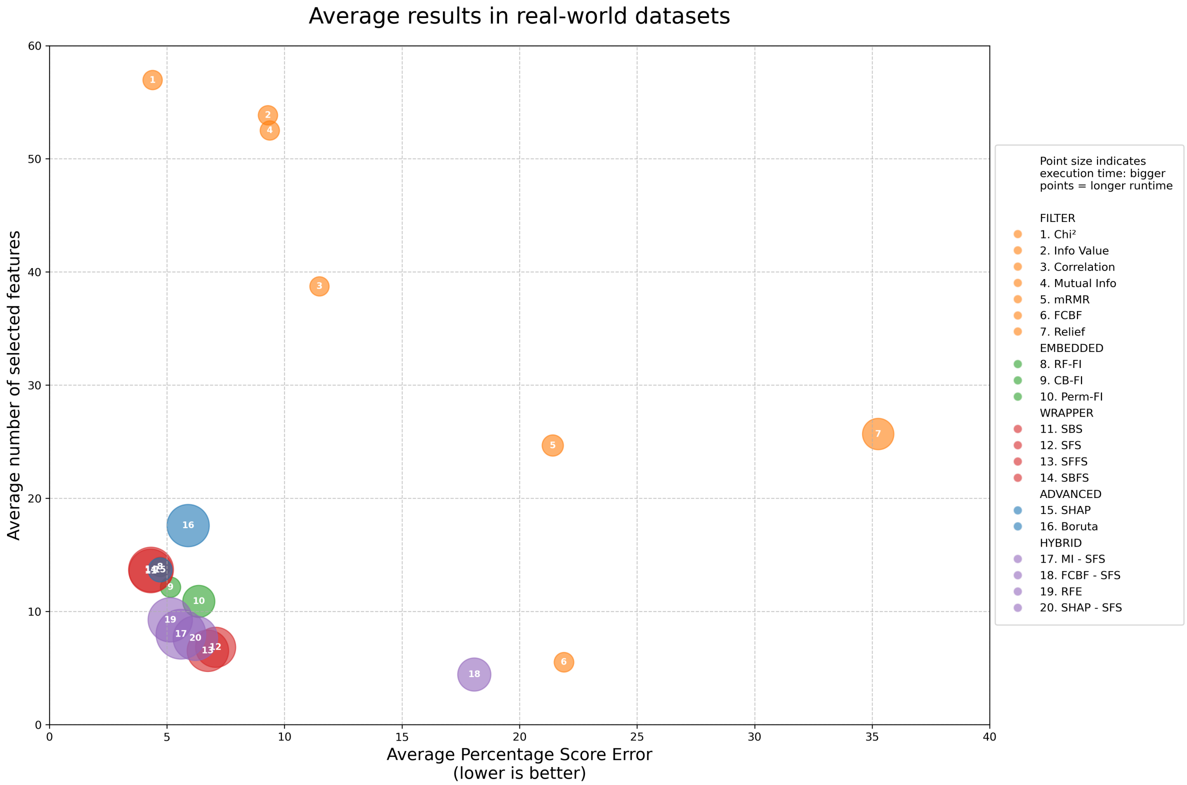

Figure 3: Average results in real-world datasets. Numerical and structured results available in Table **13** of Appendix

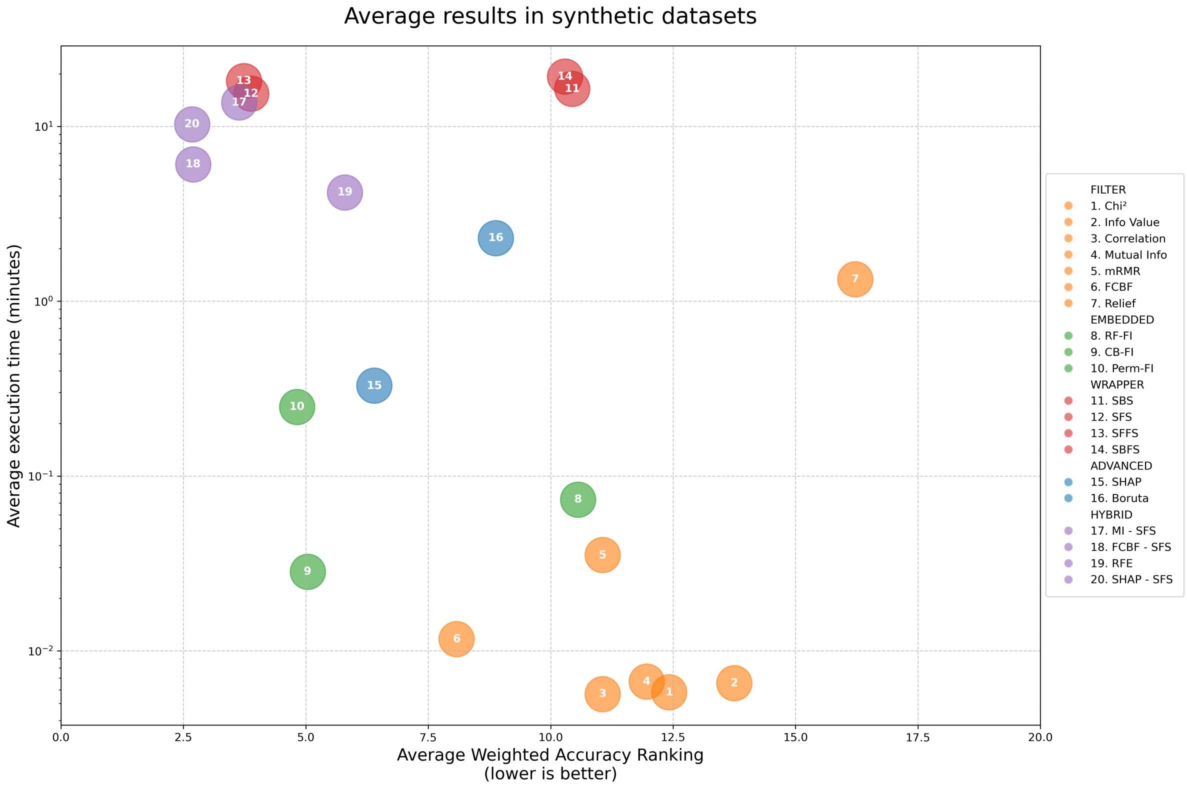

Figure 4: Average results in synthetic datasets. Numerical and structured results available in Table **14** of Appendix

References

Analytics Vidhya. (2024a). *Distinguish between tree-based machine learning models.* Retrieved from https://www.analyticsvidhya.com/blog/2021/04/distinguish-between-tree-based-machine-learning-algorithms/

Analytics Vidhya. (2024b). *Recursive feature elimination (RFE): Working, advantages and examples.* Retrieved 2024-03-28, from https://www.analyticsvidhya.com/blog/2023/05/recursive-feature-elimination/

Berrar, D., et al. (2019). *Cross-validation.*

Bobbit, Z. (2022). *What is tabular data? (definition and example).* Retrieved from https://www.statology.org/tabular-data/

Brain_Boost. (2024). *Statistics: Point-biserial correlation simply explained.* Retrieved from https://medium.com/@Brain_Boost/statistics-point-biserial-correlation-simply-explained-ce38c97b3a74

Crespo Márquez, A. (2022). The curse of dimensionality. In *Digital maintenance management: Guiding digital transformation in maintenance* (pp. 67–86). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-030-97660-6_7 doi: 10.1007/978-3-030-97660-6_7

DATAtab. (2024). *Spearman's rank correlation coefficient.* Retrieved from https://datatab.net/tutorial/spearman-correlation

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Duncan, T. E. (1970). On the calculation of mutual information. *SIAM Journal on Applied Mathematics*, *19*(1), 215–220.

Estévez, P. A., Tesmer, M., Perez, C. A., & Zurada, J. M. (2009). Normalized mutual information feature selection. *IEEE Transactions on neural networks*, *20*(2), 189–201.

Evidently AI Team. (2024). *How to explain the roc curve and roc auc score?* Retrieved from https://www.evidentlyai.com/classification-metrics/explain-roc-curve#:~:text=The%20ROC%20AUC%20score%20can,inadequate%20for%20any%20real%20applications.

Fritz. (2023a). *Hands-on with feature selection techniques: Embedded methods.* Retrieved from https://fritz.ai/hands-on-with-feature-selection-techniques-embedded-methods/

Fritz. (2023b). *Hands-on with feature selection techniques: Filter methods.* Retrieved from https://fritz.ai/hands-on-with-feature-selection-techniques-filter-methods/

Fritz. (2023c). *Hands-on with feature selection techniques: Wrapper methods part 3: Forward feature selection, backward feature elimination, exhaustive feature selection, and bidirectional search.* Retrieved from https://fritz.ai/feature-selection-techniques-wrapper-methods/

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, *35*, 507–520.

Jim, F. (2023). *Root mean square error (rmse).* Retrieved from https://statisticsbyjim.com/regression/root-mean-square-error-rmse/

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

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. In *European conference on machine learning* (pp. 171–182).

Kumar, V., & Minz, S. (2014). Feature selection. *SmartCR*, *4*(3), 211–229.

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the boruta package. *Journal of statistical software*, *36*, 1–13.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Li, Y., Mansmann, U., Du, S., & Hornung, R. (2022). Benchmark study of feature selection strategies for multi-omics data. *BMC bioinformatics*, *23*(1), 412.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, *17*(4), 491–502.

Long, F., Peng, H., & Ding, C. (2005, August). Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy . *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *27*(08), 1226-1238. Retrieved from https://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.159 doi: 10.1109/TPAMI.2005.159

Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems*, *26*.

Lundberg, S. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., … Lee, S.-I. (2020). From local explanations to global understanding with explain-able ai for trees. *Nature machine intelligence*, *2*(1), 56–67.

Medium. (2019). *Chi-square test for feature selection in machine learning.* Retrieved from `https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223`

Medium. (2020). *Hands-on with feature selection techniques: Hybrid methods.* Retrieved from `https://medium.com/@mxcsyounes/hands-on-with-feature-selection-techniques-hybrid-methods-b93b1b06d3a5`

Medium. (2021a). *Hands-on with feature selection techniques: Wrapper methods.* Retrieved from `https://medium.com/@mxcsyounes/hands-on-with-feature-selection-techniques-wrapper-methods-5bb6d99b1274`

Medium. (2021b). *Weight of evidence (woe) and information value (iv) — how to use it in eda and model building?* Retrieved from `https://anikch.medium.com/weight-of-evidence-woe-and-information-value-iv-how-to-use-it-in-eda-and-model-building-3b3b98efe0e8`

Medium. (2024). *Understanding mean absolute error (mae) in regression: A practical guide.* Retrieved from `https://medium.com/@m.waqar.ahmed/understanding-mean-absolute-error-mae-in-regression-a-practical-guide-26e80ebb97df`

Moral, M. (2025). Benchmark of feature selection techniques for tabular data. *Universitat Autònoma de Barcelona.* Retrieved from `https://github.com/miguelmoralh/feature_selection_benchmark`

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, *31*.

Pudjihartono, N., Fadason, T., Kempa-Liehr, A. W., & O'Sullivan, J. M. (2022). A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, *2*, 927312.

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, *53*, 23–69.

Sánchez-Maroño, N., Alonso-Betanzos, A., & Tombilla-Sanromán, M. (2007). Filter methods for feature selection–a comparative study. In *International conference on intelligent data engineering and automated learning* (pp. 178–187).

Shmuel, A., Glickman, O., & Lazebnik, T. (2024). A comprehensive benchmark of machine and deep learning across diverse tabular datasets. *arXiv preprint arXiv:2408.14817*.

Uddin, S., & Lu, H. (2024). Confirming the statistically significant superiority of tree-based machine learning algorithms over their counterparts for tabular data. *Plos one*, *19*(4), e0301541.

Ye, H.-J., Liu, S.-Y., Cai, H.-R., Zhou, Q.-L., & Zhan, D.-C. (2024). A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*.

Ying, X. (2019). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022).

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (icml-03)* (pp. 856–863).

Yu, T., & Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*.

8 Appendix

8.1 Feature selection techniques

Table 1: Overview of feature selection techniques

Family	**Type**	**Name**	**Description**
Filter	**Bivariate**	Correlation (DATAtab, 2024) (Brain_Boost, 2024)	Uses Spearman in regression and point-biserial in binary classification correlation coefficients to select features based on their strength of relationship with target.
		Chi-squared Test (Medium, 2019)	Uses chi-squared statistical test to identify and select significant features based on their p-values in relation to the target variable.
		Mutual Information (MI) (Duncan, 1970) (Estévez, Tesmer, Perez, & Zurada, 2009)	Selects features based on their normalized mutual information score with target.
		Information Value (Medium, 2021b)	Evaluates predictive power of features using weight of evidence and information value metrics specifically for binary classification.
	Multivariate	Minimum Redundancy Maximum Relevance (mRMR) (Long, Peng, & Ding, 2005)	Maximizes relevance (mutual information with target) while minimizing redundancy between selected features using normalized mutual information.
		Fast Correlation-Based Filter (FCBF) (L. Yu & Liu, 2003)	Uses Symmetrical Uncertainty (SU) to measure feature-target relevance and feature-feature redundancy, removing features with low SU with the target and those redundant to stronger features.
		Relief algorithms (Kononenko, 1994) (Robnik-Šikonja & Kononenko, 2003)	A comprehensive implementation of Relief family algorithms (Relief, ReliefF, RReliefF) that automatically selects and applies the appropriate variant based on the task. It evaluates features based on their ability to distinguish between similar instances by finding k-nearest neighbors for each data point and analyzing how feature values differ between neighbors.
Embedded	**Importance**	Random Forest Feature Importance (Louppe, Wehenkel, Sutera, & Geurts, 2013)	Uses Random forest mean decrease in impurity (MDI) importance metric to select features based on their average contribution to node splits.
		CatBoost Feature Importance (Prokhorenkova, Gusev, Vorobev, Dorogush, & Gulin, 2018)	Selects features based on CatBoost's LossFunctionChange importance metric, which measures each feature's contribution to model loss reduction.
		Permutation Feature Importance	Selects features by measuring model performance drop when each feature's values are randomly permuted, using cross-validation for robust importance estimation.
Wrapper (Fritz, 2023c)	**Backward**	Sequential Backward Selection (SBS)	Iteratively removes features based on cross-validated model performance, continues until performance drops exceed threshold.
	Forward	Sequential Forward Selection (SFS)	Iteratively adds features based on cross-validated model performance, continues until no feature addition improves score beyond minimum threshold.
	Bidirectional	Sequential Forward Floating Selection (SFFS)	Extends SFS by adding floating phase - after adding feature, evaluates if removing previously selected features improves performance.

Continues on next page...

Table 1 – Continued from previous page

Family	Type	Name	Description
		Sequential Backward Floating Selection (SBFS)	Extends SBS by adding floating phase - after removing feature, evaluates if returning previously removed features improves performance.
Advanced		SHAP (S. Lundberg, 2017) (S. M. Lundberg et al., 2020)	Uses SHAP (SHapley Additive exPlanations) values to quantify the contribution of each feature to model predictions through cross-validation. Features with normalized SHAP values above a specified threshold are retained.
		Boruta (Kursa & Rudnicki, 2010)	Compares original feature importance scores against shadow (randomized) features across multiple iterations to statistically determine which features are truly relevant.
Hybrid (Liu & Yu, 2005) (Kohavi & John, 1997)	**Filter - wrapper**	MI - SFS	Applies bivariate normalized mutual information with a permissive threshold for initial filtering, then uses SFS to fine-tune the selection.
		FCBF - SFS	Uses multivariate FCBF with a permissive threshold for initial filtering, followed by SFS to optimize the final feature subset.
	Embedded - wrapper	Recursive Feature Elimination (RFE) (Analytics Vidhya, 2024b)	Iteratively removes features based on importance scores while monitoring cross-validated performance to find minimal effective subset.
	Advanced - wrapper	SHAP - SFS	Applies advanced SHAP with a permissive threshold for initial selection, followed by SFS to optimize the reduced feature subset.

8.2 Datasets description

You can find the real-world datasets sources by clicking on the name of the datasets.

Table 2: Datasets Summary

ID	Dataset	Task	Classes	Categorical	Numerical	Imbalanced	Samples	Domain
1	breast-cancer-wisconsin-data	Clas	2	0	31	No	569	Healthcare
2	ailerons	Reg		0	40		13750	Industrial
3	amazon_employee_access	Clas	2	9	0	Yes	32769	Financial
4	diamonds	Reg		3	6		53940	Natural
5	phpYLeydd	Clas	5	0	32	No	9873	Vision
6	indian_pines	Clas	8	0	220	No	9144	Physical
7	WEC_Perth_49	Reg		0	148		36043	Natural
8	phpV5QYya	Clas	2	30	0	No	11055	Leisure
9	weatherAUS	Clas	2	6	16	No	145460	Natural
10	allbp	Clas	3	23	6	Yes	3772	Biological
11	car-evaluation	Clas	4	21	0	Yes	1728	Physical
12	dataset_23_cmc	Clas	3	7	2	No	1473	Healthcare
13	data	Clas	2	0	95	Yes	6819	Financial
14	dataset_2183_cpu_act	Reg		0	21		8192	Computer
15	Invistico_Airline	Clas	2	3	18	No	129880	Industrial
16	dna	Clas	3	180	0	No	3186	Biological
17	ObesityDataSet_raw_and_data_sinthetic	Clas	7	8	8	No	2111	Healthcare
18	insurance	Reg		3	3		1338	Healthcare
19	dataset_2204_house_8L	Reg		0	8		22784	Societal
20	jungle_chess_2pcs_raw_endgame_complete	Clas	3	0	6	No	44819	Game
21	dataset_2175_kin8nm	Reg		0	8		8192	Physical

Continues on next page...

Table 2 – Continued from previous page

ID	Dataset	Task	Classes	Categorical	Numerical	Imbalanced	Samples	Domain
22	php7zhUPY	Clas	18	3	3	No	28056	Game
23	dataset_20_mfeat-pixel	Clas	10	240	0	No	2000	Vision
24	ozone_level	Clas	2	72	0	Yes	2536	Natural
25	seismic-bumps	Clas	2	4	14	Yes	2584	Natural
26	dataset	Clas	2	17	3	No	7043	Financial
27	php2PPvrr	Clas	5	20	6	No	2800	Healthcare
28	topo_2_1	Reg		0	266		8885	Natural
29	php0iVrYT	Clas	2	0	4	No	748	Societal
30	dataset_50_tic-tac-toe	Clas	2	9	0	No	958	Game
31	php8Mz7BG	Clas	2	0	5	No	5404	Physical
32	data_alzheimer	Clas	2	1	450	No	174	Healthcare
33	phpDYCOet	Clas	2	29	89	No	34465	Computer
34	php88ZB4Q	Clas	6	0	561	No	10299	Physical
35	phpeZQVCe	Reg		1	126		1994	Societal
36	ds_salaries	Reg		7	3		3755	Financial
37	dataset_customer	Clas	2	4	9	Yes	1723	Financial
38	cerebral_stroke	Clas	2	5	6	Yes	43400	Healthcare
39	Credit	Clas	2	4	4	Yes	4521	Financial
40	dataset_claim	Reg		110	14		188318	Financial
41	irish	Clas	2	3	2	No	500	Educational
42	phpnYQXoc	Clas	4	6	2	No	155	Natural
43	dataset_2189_lowbwt	Reg		7	2		189	Science
44	dataset_lisbon_house	Reg		7	9		246	Financial
45	file639340bd9ca9	Reg		4	5		166821	Financial
46	online_gaming_behavior_dataset	Clas	3	4	8	No	40034	Game
47	speeddating	Clas	2	63	59	Yes	8378	Societal
48	dermatology_database_1	Clas	6	1	33	No	366	Healthcare
49	dataset_game	Clas	2	116	0	No	102944	Game
50	phpVDlhKL	Clas	2	229	0	No	64	Computer
51	dataset_1	Reg		0	4		16795	Synthetic
52	dataset_2	Reg		0	104		17850	Synthetic
53	dataset_3	Reg		0	9		42090	Synthetic
54	dataset_4	Reg		0	8		6311	Synthetic
55	dataset_5	Reg		0	106		19942	Synthetic
56	dataset_6	Clas	4	0	117	Yes	1189	Synthetic
57	dataset_7	Clas	7	0	9	No	12394	Synthetic
58	dataset_8	Clas	3	0	46	No	15502	Synthetic
59	dataset_9	Clas	5	0	113	Yes	24897	Synthetic
60	dataset_10	Reg		0	38		18159	Synthetic
61	dataset_11	Reg		0	22		20457	Synthetic
62	dataset_12	Reg		0	56		7873	Synthetic
63	dataset_13	Clas	3	0	44	No	34763	Synthetic
64	dataset_14	Clas	2	0	8	No	24247	Synthetic
65	dataset_15	Reg		0	81		22271	Synthetic
66	dataset_16	Clas	5	0	90	No	9571	Synthetic
67	dataset_17	Reg		0	9		3695	Synthetic
68	dataset_18	Reg		0	10		13666	Synthetic
69	dataset_19	Reg		0	127		39952	Synthetic
70	dataset_20	Clas	2	0	108	Yes	36222	Synthetic
71	dataset_21	Clas	4	0	57	No	43530	Synthetic
72	dataset_22	Clas	2	0	37	Yes	14773	Synthetic
73	dataset_23	Clas	2	0	9	No	1206	Synthetic
74	dataset_24	Reg		0	8		49702	Synthetic
75	dataset_25	Clas	4	0	5	Yes	46151	Synthetic
76	dataset_26	Clas	7	0	99	No	7374	Synthetic
77	dataset_27	Clas	3	0	110	Yes	34827	Synthetic
78	dataset_28	Reg		0	125		44689	Synthetic
79	dataset_29	Clas	2	0	45	No	3693	Synthetic

Continues on next page...

Table 2 – Continued from previous page

ID	Dataset	Task	Classes	Categorical	Numerical	Imbalanced	Samples	Domain
80	dataset_30	Reg		0	129		19047	Synthetic
81	dataset_31	Clas	4	0	6	Yes	2306	Synthetic
82	dataset_32	Clas	4	0	69	No	10474	Synthetic
83	dataset_33	Reg		0	42		33562	Synthetic
84	dataset_34	Clas	8	0	8	Yes	8994	Synthetic
85	dataset_35	Clas	5	0	62	No	4304	Synthetic
86	dataset_36	Reg		0	116		45417	Synthetic
87	dataset_37	Clas	2	0	17	Yes	28728	Synthetic
88	dataset_38	Clas	2	0	125	No	39623	Synthetic
89	dataset_39	Clas	3	0	8	Yes	39559	Synthetic
90	dataset_40	Clas	5	0	8	No	33970	Synthetic
91	dataset_41	Clas	2	0	121	Yes	38157	Synthetic
92	dataset_42	Clas	5	0	10	No	1197	Synthetic
93	dataset_43	Clas	2	0	69	Yes	23671	Synthetic
94	dataset_44	Clas	2	0	116	No	7546	Synthetic
95	dataset_45	Clas	2	0	127	Yes	12411	Synthetic
96	dataset_46	Clas	2	0	10	No	36153	Synthetic
97	dataset_47	Clas	2	0	5	Yes	9120	Synthetic
98	dataset_48	Clas	2	0	119	No	47843	Synthetic
99	dataset_49	Clas	2	0	13	Yes	32616	Synthetic
100	dataset_50	Clas	2	0	107	No	44016	Synthetic

8.3 Datasets distribution and limitations

The dataset collection process encountered several challenges that impacted the balanced representation of certain dataset categories:

- High-dimensional datasets: Obtaining real-world datasets with more than 100 features proved particularly challenging. Many public available repositories primarily contain lower dimensional datasets, limiting the representation of high-dimensional scenarios.
- Imbalanced datasets: Highly imbalanced datasets, especially in classification tasks, are less common in public repositories. This scarcity made it difficult to achieve a perfectly balanced distribution of class imbalance across the real-world dataset collection.
- Primarily categorical and mixed-type datasets: For the synthetic datasets, all features were intentionally generated as numerical. This decision was made to ensure clear and controlled mathematical relationships between features and target variables, allowing for precise feature selection method evaluation. Consequently, primarily categorical and mixed-type datasets appear less represented.

These constraints highlight the ongoing challenges in creating truly representative dataset collections for machine learning research. Here ypu can find the dataset distribution of characteristics across the 100 datasets:

Table 3: Dimensionality summary

	High-dimensional	**Mid-dimensional**	**Low-dimensional**
Number of datasets	29	37	34

Table 4: Task summary

	Regression	**Binary classification**	**Multiclass classification**
Number of datasets	31	37	32

Table 5: Features type summary

	Primarily numerical	Primarily categorical	Mixed-type
Number of datasets	68	11	21

Table 6: Target distribution summary

	Balanced	Imbalanced
Number of datasets	42	27

8.4 Statistical distributions of base data in synthetic datasets

Table 7: NumPy random distributions and parameters

Distribution	Parameters	Values
numpy.random.normal	loc	0
	scale	1
numpy.random.lognormal	mean	0
	sigma	0.5
numpy.random.exponential	scale	1.0
numpy.random.beta	a	2
	b	5
numpy.random.gamma	shape	2
	scale	2
numpy.random.weibull	a	1.5
numpy.random.chisquare	df	3

8.5 Method restrictions

The application of certain feature selection methods was restricted based on two types of constraints:

8.5.1 Computational restrictions

To maintain computational feasibility while ensuring comprehensive evaluation:

- Wrapper methods (Sequential Forward, Backward and Floating variants) were limited to datasets with fewer or equal than 100 features (high-dimensional datasets) due to their iterative nature requiring multiple model training cycles for each feature evaluation.

These restrictions were implemented based on empirical testing that showed exponential increases in computation time for these methods as feature dimensionality grew.

8.5.2 Method-specific applicability restrictions

Some methods have inherent limitations in their applicability:

- Information Value method was restricted to only binary classification tasks due to its underlying mathematical formulation being specific to binary outcomes.

- Correlation method was not applied to multiclass classification problems as it is designed for binary classification and regression tasks.

8.6 Cross-validation strategies

This study employs different cross-validation strategies depending on the task type:

- For regression tasks, k-fold cross-validation with k=3 is used. The data is split into 3 equal-sized folds, and the model is trained on 2 folds and validated on the remaining fold. This process is repeated 3 times, with each fold serving as the validation set once. The validation scores are then averaged to provide a robust estimate of model performance.
- For classification tasks (both binary and multiclass), stratified k-fold cross-validation with k=3 is used. This variant ensures that the class distribution in each fold is representative of the overall class distribution in the dataset. Like in the regression case, the model is trained on 2 folds and validated on the remaining fold, with the process repeated 3 times. Stratification is particularly important for imbalanced datasets to ensure that the model is exposed to a sufficient number of instances from the minority class or classes during training and validation.

8.7 Feature selection methods thresholds

Table 8: Thresholds used in feature selection methods

Method	**Threshold Value**	**Description**
Chi-squared Test	0.05	Statistical significance level (p-value). Features with p-values below this threshold are considered significant.
Information Value	0.02	Features with information value above this threshold are considered predictive. Standard threshold based on WOE/IV analysis guidelines. Raw IV scores, not normalized.
Correlation	0.1	Features with absolute correlation above this threshold are selected. Applied to absolute correlation values. Raw correlation scores in range [-1,1].
Mutual Information	0.01	Features with normalized mutual information score above this threshold are selected.
mRMR	0.01	Features with mRMR score above this threshold are selected. The score (normalized to [-1,1]) represents a balance between relevance (normalized mutual information with target) and redundancy (normalized mutual information with other features).
FCBF	0.01	First, features with Symmetrical Uncertainty (SU) above this threshold are selected. Then remaining features are retained or removed accoring to FCBF criterion 'A feature Fj is redundant to Fi if SU(Fi,Fj) >= SU(Fj,target)'
Relief algorithms	0.1	Features with Relief score above this threshold (normalized to [-1,1]) are selected. Scores represent how well features distinguish between nearby instances. Positive scores indicate the feature helps distinguish between different classes/values, negative scores suggest the feature might be noisy.
Random Forest Feature Importance	0.01	Features with normalized importance score above this threshold (normalized to [0,1]) are selected. Raw importance scores are normalized to sum to 1

Continues on next page...

Table 8 – Continued from previous page

Method	Threshold Value	Description
CatBoost Feature Importance	0.01	Features with normalized importance score above this threshold (normalized to [0,1]) are selected. Raw importance scores are normalized to sum to 1
Permutation Feature Importance	0.01	Features with normalized Permutation Importance above this threshold (normalized to [0,1]) are selected. Raw importance scores are normalized to sum to 1
Sequential Backward Selection	0.001	Maximum allowed relative performance drop when removing a feature (e.g., 0.001 means 0.1% drop allowed).
Sequential Forward Selection	0.001	Minimum required relative performance improvement when adding a feature (e.g., 0.001 means 0.1% improvement required).
Sequential Forward Floating Selection	0.001	Minimum required performance improvement for both addition and removal phases (0.1%).
Sequential Backward Floating Selection	0.001	Maximum allowed performance drop for both removal and addition phases (0.1%).
SHAP	0.01	Features with normalized SHAP values above this threshold (normalized to [0,1]) are selected. Raw SHAP values are normalized to sum to 1.
Boruta	0.05	Statistical significance level for comparing original feature importance with shadow (random) feature importances. Features with p-values below this threshold are retained.
MI - SFS	0.001 for both	normalized mutual information threshold for initial filtering and minimum improvement for SFS phase.
FCBF - SFS	0.001 for both	FCBF threshold for initial filtering and minimum improvement for SFS phase.
Recursive Feature Elimination	0.015	Maximum allowed relative performance drop (1.5%) from best score. All the iterations are executed.
SHAP - SFS	0.001 for both	SHAP threshold for initial filtering and minimum improvement for SFS phase.

8.8 Hyperparameters

Table 9: CatBoost hyperparameters for regression tasks

Parameter	**Value**
iterations	200
depth	6
learning_rate	0.1
loss_function	RMSE
bootstrap_type	Bernoulli
subsample	0.8
early_stopping_rounds	30
eval_fraction	0.2
l2_leaf_reg	1.0
random_strength	0.5
min_data_in_leaf	n_samples * 2/3 * 0.01

Table 10: CatBoost hyperparameters for binary classification tasks

Parameter	**Value**
iterations	200
depth	6
learning_rate	0.1
loss_function	Logloss
bootstrap_type	Bernoulli
subsample	0.8
early_stopping_rounds	30
eval_fraction	0.2
l2_leaf_reg	1.0
random_strength	0.5
min_data_in_leaf	n_samples * 2/3 * 0.01

Table 11: CatBoost hyperparameters for multiclass classification tasks

Parameter	**Value**
iterations	200
depth	6
learning_rate	0.1
loss_function	MultiClass
bootstrap_type	Bernoulli
subsample	0.8
early_stopping_rounds	30
eval_fraction	0.2
l2_leaf_reg	1.0
random_strength	0.5
min_data_in_leaf	n_samples * 2/3 * 0.01

Table 12: Random Forest hyperparameters

Parameter	**Value**
n_estimators	100
max_depth	6
max_features	sqrt
min_samples_leaf	n_samples * 2/3 * 0.01

8.9 Real-world datasets prepocessing

The considered datasets do not require complex or highly specific data cleaning. Initial preprocessing is applied to all real-world datasets, where constant features (features with only one unique value along all data points) are removed. This is done to avoid problems when executing some feature selection techniques. Moreover, in classification tasks it is ensured that the target is numerical (using LabelEncoder) and that in binary classification imbalanced cases the minority class is the positive one (needed for Average Precision score). Finally, specific cleanings were required for certain datasets:

- *speeddating* real-world dataset: removed 'decision' and 'decision_o' columns to prevent data leakage

- *weatherAUS* real-world dataset: removed samples with missing target values

- *irish* real-world dataset: removed 'Educational_level' column to prevent data leakage.

As CatBoost internally manages missing values and categorical features, there is no needed preprocessing before evaluating on the dataframe of the selected features by the methods.

8.10 Result tables

Table 13: Average metrics for real-world datasets

Family	Method	Average Percentage Score Error	Average number of selected features	Average execution time (min)
Filter	Chi²	4.38	56.96	0.007
	Correlation	11.49	38.74	0.004
	Info Value	9.29	53.86	0.008
	Mutual Info	9.38	52.52	0.005
	mRMR	21.40	24.66	0.158
	FCBF	21.88	5.52	0.029
	Relief	35.25	25.68	2.019
Embedded	RF-FI	4.71	13.94	0.013
	CB-FI	5.14	12.14	0.083
	Perm-FI	6.35	10.90	2.277
Wrapper	SBS	4.31	13.59	14.942
	SFS	7.06	6.84	8.367
	SFFS	6.73	6.54	10.407
	SBFS	4.31	13.70	18.369
Advanced	SHAP	4.70	13.68	0.456
	Boruta	5.90	17.60	11.839
Hybrid	MI - SFS	5.59	8.00	43.397
	FCBF - SFS	18.07	4.44	2.706
	RFE	5.14	9.26	16.759
	SHAP - SFS	6.21	7.64	17.196

Table 14: Average results for synthetic datasets

Family	Method	Average Weighted Accuracy Ranking	Average execution time (min)
Filter	Chi²	12.42	0.006
	Correlation	11.06	0.006
	Info Value	13.75	0.007
	Mutual Info	11.96	0.007
	mRMR	11.06	0.035
	FCBF	8.08	0.012
	Relief	16.22	1.336
Embedded	RF-FI	10.56	0.073
	CB-FI	5.04	0.028
	Perm-FI	4.82	0.249
Wrapper	SBS	10.44	16.407
	SFS	3.88	15.404
	SFFS	3.73	18.189
	SBFS	10.29	19.267

Continues on next page...

Table 14 – Continued from previous page

Family	Method	Average Weighted Accuracy Ranking	Average Execution Time (min)
Advanced	SHAP	6.4	0.329
	Boruta	8.88	2.294
Hybrid	MI - SFS	3.64	13.701
	FCBF - SFS	2.7	6.065
	RFE	5.80	4.193
	SHAP - SFS	2.68	10.267

8.11 Detailed results analysis

While the previous general analysis provides valuable overall insights, examining performance patterns across specific dataset characteristics reveals additional understanding of method effectiveness. The following sections present detailed analysis of method performance across different dimensionalities and task types. While analysis of additional data characteristics like class balance and feature types would offer further insights, this section focuses on dimensionality and task type as they are considered the most influential factors in feature selection.

8.11.1 Dimensionality results analysis

The following analysis examines the most effective feature selection techniques for each dimensionality category, focusing on methods that achieve optimal results in relation to performance, selectivity and computational efficiency. Detailed results can be found in Appendix **8.12**.

- **High-dimensional datasets (>100 features)**: Advanced and hybrid methods demonstrate superior performance but with significant differences in computational efficiency. SHAP achieves excellent performance in both scenarios (synthetic ranking 4.06, real-world error 3.93%) with reasonable execution time (0.349/0.705 minutes) and moderate selectivity (19.92 features). Hybrid methods show outstanding performance but at substantial computational cost, SHAP - SFS (synthetic ranking 2.44, real-world error 3.39%, 10.69 features) and RFE (synthetic ranking 4.63, real-world error 3.06%, 13.31 features). Among faster methods, CatBoost Feature Importance offers balanced performance (synthetic ranking 5.81, real-world error 7.19%) with efficient execution (0.033/0.174 minutes) though lower selectivity (19.92 features). While Boruta achieves the lowest error rate (1.46%), its poor selectivity (42.77 features) and high execution time (28.285 minutes) make it less practical. Filter methods generally struggle, with most showing high error rates and/or poor selectivity, though FCBF achieves decent synthetic ranking (9.69) with good selectivity (10.77 features).
- **Mid-dimensional datasets (11-100 features)**: Embedded methods demonstrate exceptional effectiveness. CatBoost Feature Importance achieves strong performance across both scenarios (synthetic ranking 6.24, real-world error 4.98%) while maintaining excellent selectivity (12.05 features) and fast execution (0.029/0.055 minutes). Permutation Feature Importance shows similar strength (synthetic ranking 6.12, real-world error 6.14%, 11.80 features) but requires more computation time (0.202/0.262 minutes). Forward wrapper methods excel in synthetic scenarios (rankings 4.82/4.53 for SFS/SFFS) with good real-world performance (8.30/8.05% error) and strong selectivity (8.35/8.00 features), though at significant computational cost (13-16 minutes). Their backward counterparts show slightly better error rates (6.85%) but poorer selectivity (19.75 features) and longer execution times. Hybrid methods demonstrate mixed results; RFE achieves good balance (synthetic ranking 7.24, real-world error 6.91%, 10.35 features, 3.78/3.58 minutes), while SHAP - SFS and FCBF - SFS excel in synthetic scenarios (rankings 3.41/3.00) but with higher computational overhead.
- **Low-dimensional datasets ($\leq$10 features)**: Embedded methods work particularly well, with Permutation Feature Importance achieving exceptional performance across both scenarios (synthetic ranking 2.00, real-world error 3.84%) with excellent selectivity (6.18 features) and reasonable execution time (0.064/0.162 minutes). CatBoost Feature Importance shows similar strength (synthetic ranking 3.12, real-world error 3.76%, 6.29 features) with even faster execution (0.023/0.046 minutes). Wrapper methods become highly practical, with forward approaches achieving excellent synthetic rankings (2.94) and reasonable error rates (5.94-6.29%) with good selectivity (4.82-5.06 features) and manageable execution times (1-3.5 minutes). Hybrid methods maintain strong performance; SHAP+SFS and RFE achieve good error rates (4.30/4.64%) and selectivity (5.06/4.88 features). Information Value stands out in binary classification tasks with remarkable performance (0.73% error, 6.33 features) but limited applicability to binary classification tasks.

8.11.2 Task results analysis

The following section examines the most effective feature selection methods across different types of machine learning tasks. Complete results for each task type can be found in Appendix **8.13**.

- **Regression tasks**: Several methods demonstrate strong performance across both synthetic and real-world scenarios. SHAP - SFS excels with impressive synthetic ranking (3.00) and real-world performance (4.68% error, 8.29 features), though requiring significant computation time (20.172/9.638 minutes). RFE shows similar strength in both contexts (synthetic ranking 6.47, real-world error 3.39%) with good selectivity and slightly lower execution times (13.544/4.118 minutes). FCBF - SFS achieves excellent synthetic ranking (3.06) but struggles in real-world scenarios (29.73% error). Among faster methods, SHAP demonstrates consistent effectiveness (synthetic ranking 3.82, real-world error 5.33%, 12.21 features) with efficient execution (0.317/0.461 minutes). Forward wrapper methods perform well in both contexts (synthetic rankings 6.55-6.64, real-world errors 3.29-4.44%) with excellent selectivity (6.40-6.80 features), though at higher but manageable computational cost (5-6 minutes).
- **Binary classification**: Several methods show strong performance across both scenarios. Embedded methods excel, with CatBoost Feature Importance achieving strong real-world performance (5.65%) and synthetic ranking (4.50) while maintaining good selectivity (13.52) and low execution times for CatBoost (0.025/0.110). Forward wrapper methods demonstrate exceptional synthetic accuracy (ranking 1.33) desite its poor real-world performance (error 11.09%). The backward variants show better real-world error (6.20%) but retain more features (18.50). Hybrid methods show excellent synthetic performance, particularly SHAP - SFS (ranking 2.13), though with substantial execution times (11-23 minutes). SHAP maintains consistent performance across both scenarios (synthetic ranking 6.13, real-world error 6.08%) with efficient execution (0.325/0.512 minutes).
- **Multiclass classification**: Embedded methods demonstrate remarkable effectiveness. CatBoost Feature Importance and Permutation Feature Importance achieve outstanding real-world performance (1.45%, 1.48% error) and strong synthetic rankings (4.82, 5.53) while maintaining excellent selectivity (12.00, 9.93 features). CatBoost particularly impresses with its computational efficiency (0.036/0.045 minutes). Hybrid methods such as SHAP - SFS and RFE achieve excellent error rates (2.96%, 2.96%) and synthetic rankings (2.88, 4.65), though requiring substantial computation time (10-18 minutes). Forward wrapper methods maintain strong synthetic rankings (3.07-3.36) and real-world performance (3.52-3.60% error, 7.36-8.00 features) despite their elevated computation time (19-22 minutes). Chi-squared achieves impressive error rate (1.77%) but extremely poor selectivity (88.07 features), demonstrating the importance of considering both metrics.

8.12 Dimensionality result tables

Table 15: Average metrics by dimensionality for real-world datasets

Family	Dimensionality	Method	Average Percentage Score Error	Average number of selected features	Average execution time (min)
Filter	High	Chi²	0.96	177.54	0.020
		Correlation	13.64	114.44	0.013
		Info Value	22.32	161.20	0.020
		Mutual Info	9.52	168.00	0.015
		mRMR	8.41	80.23	0.592
		FCBF	16.14	10.77	0.087
		Relief	10.64	76.08	5.800
	Mid	Chi²	7.49	21.40	0.003
		Correlation	9.89	19.46	0.001
		Info Value	7.91	28.70	0.006
		Mutual Info	7.63	17.65	0.002

Continues on next page...

Table 15 – Continued from previous page

Family	Dimensionality	Method	Average Percentage Score Error	Average number of selected features	Average execution time (min)
		mRMR	22.82	7.40	0.008
		FCBF	21.43	4.70	0.009
		Relief	38.70	12.05	1.043
	Low	Chi²	3.33	6.59	0.001
		Correlation	11.60	5.62	0.001
		Info Value	0.73	6.33	0.002
		Mutual Info	11.32	5.24	0.001
		mRMR	29.68	2.47	0.002
		FCBF	26.78	2.47	0.008
		Relief	50.02	3.18	0.276
Embedded	High	RF-FI	4.39	21.08	0.025
		CB-FI	7.19	19.92	0.174
		Perm-FI	9.96	15.69	8.144
	Mid	RF-FI	4.77	15.40	0.011
		CB-FI	4.98	12.05	0.055
		Perm-FI	6.14	11.80	0.262
	Low	RF-FI	4.88	6.76	0.007
		CB-FI	3.76	6.29	0.046
		Perm-FI	3.84	6.18	0.162
Wrapper	High	N/A	N/A	N/A	N/A
	Mid	SBS	6.85	19.75	26.004
		SFS	8.30	8.35	13.475
		SFFS	8.05	8.00	16.213
		SBFS	6.85	19.95	32.244
	Low	SBS	4.06	6.35	1.928
		SFS	6.29	5.06	2.358
		SFFS	5.94	4.82	3.575
		SBFS	4.06	6.35	2.046
Advanced	High	SHAP	3.93	19.92	0.705
		Boruta	1.46	42.77	28.285
	Mid	SHAP	6.04	15.30	0.374
		Boruta	9.85	11.85	6.253
	Low	SHAP	3.71	7.00	0.363
		Boruta	4.63	5.12	5.836
Hybrid	High	MI - SFS	3.33	11.77	145.723
		FCBF - SFS	9.10	7.08	7.552
		RFE	3.06	13.31	57.500
		SHAP - SFS	3.39	10.69	47.382
	Mid	MI - SFS	8.41	8.05	11.770
		FCBF - SFS	21.13	4.10	1.464
		RFE	6.91	10.35	3.785
		SHAP - SFS	9.66	7.85	9.964

Continues on next page...

Table 15 – Continued from previous page

Family	Dimensionality	Method	Average Percentage Score Error	Average number of selected features	Average execution time (min)
	Low	MI - SFS	4.00	5.06	2.356
		FCBF - SFS	21.33	2.82	0.461
		RFE	4.64	4.88	0.867
		SHAP - SFS	4.30	5.06	2.621

Table 16: Average results by dimensionality for synthetic datasets

Family	Dimensionality	Method	Weighted Accuracy Ranking	Average Execution Time (min)
Filter	High	Chi²	11.56	0.012
		Correlation	10.92	0.011
		Info Value	11.57	0.012
		Mutual Info	11.81	0.015
		mRMR	12.06	0.087
		FCBF	9.69	0.023
		Relief	14.63	2.888
	Mid	Chi²	13.06	0.005
		Correlation	12.60	0.003
		Info Value	15.80	0.004
		Mutual Info	13.65	0.004
		mRMR	13.18	0.021
		FCBF	10.06	0.011
		Relief	17.06	1.130
	Low	Chi²	12.59	0.001
		Correlation	9.70	0.001
		Info Value	15.00	0.001
		Mutual Info	10.41	0.001
		mRMR	8.00	0.002
		FCBF	4.59	0.002
		Relief	16.88	0.082
Embedded	High	RF-FI	9.81	0.141
		CB-FI	5.81	0.033
		Perm-FI	6.44	0.495
	Mid	RF-FI	12.35	0.058
		CB-FI	6.24	0.029
		Perm-FI	6.12	0.202
	Low	RF-FI	9.47	0.025
		CB-FI	3.12	0.023
		Perm-FI	2.00	0.064
Wrapper	High	N/A	N/A	N/A
	Mid	SBS	14.35	31.439
		SFS	4.82	29.776
		SFFS	4.53	35.218
		SBFS	14.06	36.650

Continues on next page

Table 16 – Continued from previous page

Family	Dimensionality	Method	Average Weighted Accuracy Ranking	Average Execution Time (min)
	Low	SBS	6.53	1.374
		SFS	2.94	1.032
		SFFS	2.94	1.160
		SBFS	6.53	1.885
Advanced	High	SHAP	4.06	0.349
		Boruta	7.38	3.016
	Mid	SHAP	6.29	0.332
		Boruta	10.53	2.247
	Low	SHAP	8.71	0.308
		Boruta	8.65	1.662
Hybrid	High	MI - SFS	4.25	28.700
		FCBF - SFS	3.13	12.024
		RFE	4.63	8.786
		SHAP - SFS	2.44	20.989
	Mid	MI - SFS	4.12	12.692
		FCBF - SFS	3.00	6.232
		RFE	7.24	3.589
		SHAP - SFS	3.41	9.471
	Low	MI - SFS	2.59	0.593
		FCBF - SFS	2.00	0.291
		RFE	5.47	0.474
		SHAP - SFS	2.18	0.971

8.13 Task tables

Table 17: Average metrics by task type for real-world datasets

Family	Task	Method	Average Percentage Score Error	Average number of selected features	Average execution time (min)
Filter	Regression	Chi^2	2.83	50.36	0.007
		Correlation	9.16	33.71	0.008
		Mutual Info	10.79	32.93	0.008
		mRMR	32.95	7.50	0.036
		FCBF	35.73	3.43	0.016
		Relief	70.42	22.86	2.931
	Binary	Chi^2	7.28	39.14	0.007
		Correlation	13.04	42.10	0.002
		Info Value	9.29	53.86	0.008
		Mutual Info	13.68	45.62	0.004
		mRMR	21.68	13.24	0.045
		FCBF	21.85	4.19	0.025
		Relief	25.93	18.62	2.715

Continues on next page...

Table 17 – Continued from previous page

Family	Task	Method	Average Percentage Score Error	Average number of selected features	Average execution time (min)
	Multiclass	Chi^2	1.77	88.07	0.007
		Mutual Info	2.03	80.47	0.003
		mRMR	10.24	56.67	0.429
		FCBF	8.99	9.33	0.047
		Relief	15.48	38.20	0.194
Embedded	Regression	RF-FI	5.80	12.71	0.019
		CB-FI	8.34	10.21	0.082
		Perm-FI	10.58	9.57	4.021
	Binary	RF-FI	4.55	14.14	0.011
		CB-FI	5.65	13.52	0.110
		Perm-FI	7.01	12.48	2.032
	Multiclass	RF-FI	3.91	14.80	0.012
		CB-FI	1.45	12.00	0.045
		Perm-FI	1.48	9.93	0.993
Wrapper	Regression	SBS	3.21	11.60	4.475
		SFS	4.44	6.80	5.122
		SFFS	3.29	6.40	6.293
		SBFS	3.21	11.60	4.060
	Binary	SBS	6.20	18.50	23.040
		SFS	11.09	6.06	10.588
		SFFS	11.09	6.06	13.266
		SBFS	6.20	18.50	29.041
	Multiclass	SBS	2.56	8.27	12.679
		SFS	3.60	8.00	8.086
		SFFS	3.52	7.36	9.988
		SBFS	2.57	8.64	15.854
Advanced	Regression	SHAP	5.33	12.21	0.461
		Boruta	3.19	21.50	12.005
	Binary	SHAP	6.08	14.05	0.512
		Boruta	10.06	13.00	16.394
	Multiclass	SHAP	2.17	14.53	0.374
		Boruta	2.60	20.40	5.309
Hybrid	Regression	RFE	3.39	11.00	13.544
		MI - SFS	3.92	9.57	67.065
		FCBF - SFS	29.73	3.79	0.721
		SHAP - SFS	4.68	8.29	20.172
	Binary	RFE	7.85	10.24	17.697
		MI - SFS	8.67	6.95	22.217
		FCBF - SFS	16.74	4.10	2.047
		SHAP - SFS	9.54	7.33	19.880
	Multiclass	RFE	2.96	6.27	18.447
		MI - SFS	2.83	8.00	50.960
		FCBF - SFS	9.06	5.53	5.481

Continues on next page...

Table 17 – Continued from previous page

Family	**Task**	**Method**	**Average Percentage Score Error**	**Average number of selected features**	**Average execution time (min)**
		SHAP - SFS	2.96	7.47	10.661

Table 18: Average results by task type for synthetic datasets

Family	**Task**	**Method**	**Average Weighted Accuracy Ranking**	**Average execution time (min)**
Filter	Regression	Chi²	11.00	0.006
		Correlation	10.12	0.008
		Mutual Info	10.82	0.009
		mRMR	9.94	0.049
		FCBF	6.82	0.013
		Relief	16.59	1.564
	Binary	Chi²	14.00	0.006
		Correlation	12.06	0.003
		Info Value	13.75	0.007
		Mutual Info	13.31	0.008
		mRMR	13.06	0.035
		FCBF	11.19	0.011
		Relief	17.06	1.652
	Multiclass	Chi²	12.35	0.005
		Mutual Info	11.82	0.004
		mRMR	10.29	0.022
		FCBF	6.41	0.011
		Relief	15.06	0.811
Embedded	Regression	RF-FI	8.94	0.084
		CB-FI	5.76	0.024
		Perm-FI	3.94	0.217
	Binary	RF-FI	12.00	0.083
		CB-FI	4.50	0.025
		Perm-FI	5.00	0.262
	Multiclass	RF-FI	10.82	0.053
		CB-FI	4.82	0.036
		Perm-FI	5.53	0.268
Wrapper	Regression	SBS	15.82	5.774
		SFS	6.64	15.621
		SFFS	6.55	19.578
		SBFS	15.82	5.997
	Binary	SBS	4.56	18.569
		SFS	1.33	8.372
		SFFS	1.33	9.662
		SBFS	3.67	30.363
	Multiclass	SBS	10.00	23.371
		SFS	3.36	19.755
		SFFS	3.07	22.580
		SBFS	10.21	22.562

Continues on next page...

Table 18 – Continued from previous page

Family	**Task**	**Method**	**Average Weighted Accuracy Ranking**	**Average execution time (min)**
Advanced	Regression	SHAP	3.82	0.317
		Boruta	8.76	1.944
	Binary	SHAP	6.13	0.325
		Boruta	9.94	2.103
	Multiclass	SHAP	9.24	0.345
		Boruta	8.00	2.824
Hybrid	Regression	RFE	6.47	4.118
		MI - SFS	4.76	12.127
		FCBF - SFS	3.06	5.626
		SHAP - SFS	3.00	9.638
	Binary	RFE	6.31	4.139
		MI - SFS	3.25	13.499
		FCBF - SFS	2.88	5.772
		SHAP - SFS	2.13	11.475
	Multiclass	RFE	4.65	4.318
		MI - SFS	2.88	15.466
		FCBF - SFS	2.18	6.781
		SHAP - SFS	2.88	9.758