Impact of filter feature selection on classification: an empirical study

Uchechukwu F. NJOKU
Universitat Politècnica de Catalunya
unjoku@essi.upc.edu

Alberto ABELLÓ
Universitat Politècnica de Catalunya
aabello@essi.upc.edu

Besim BILALLI
Universitat Politècnica de Catalunya
bbilalli@essi.upc.edu

Gianluca BONTEMPI
Université Libre de Bruxelles
gianluca.bontempi@ulb.be

ABSTRACT

The high-dimensionality of Big Data poses challenges in data understanding and visualization. Furthermore, it leads to lengthy model building times in data analysis and poor generalization for machine learning models. Consequently, there is a need for feature selection, which allows identifying the more relevant part of the data to improve the data analysis (e.g., building simpler and more understandable models with reduced training time and improved model performance).

This study aims to (i) characterize the factors (i.e., dataset characteristics) that influence the performance of feature selection methods, and (ii) assess the impact of feature selection on the training time and accuracy of binary and multiclass classification problems. As a result, we propose a systematic method to select representative datasets (i.e., considering the distributions of several dataset characteristics) in a given repository. Next, we provide an empirical study of the impact of eight feature selection methods on Naive Bayes (NB), Nearest Neighbor (KNN), Linear Discriminant Analysis (LDA), and Multilayer Perceptron (MLP) classification algorithms using 32 real-world datasets and a relative performance measure.

We observed that feature selection is more effective in reducing training time (e.g., up to 60% for LDA classifiers) than improving classification accuracy (e.g., up to 5%). Furthermore, we observed that feature selection gives slight accuracy improvement for binary classification (i.e., up to 5%), while it mostly leads to accuracy degradation for multiclass classification. Although none of the studied feature selection methods is best in all cases, for multiclass classification, we observed that *correlation based* and *minimum redundancy maximum relevance* feature selection methods gave the best results in accuracy. Through statistical testing, we found LDA and MLP to benefit more in accuracy improvement after feature selection than KNN and NB.

1 INTRODUCTION

Nowadays, organizations store data without proper problem definition, hoping to gain some insight from the gathered data in the future. This 'blind' data collection leads to datasets that have a large number of features, often irrelevant or redundant [5] and raises challenges in data storage, understanding, visualization, model training time and explainability. In addition to these, there is a problem of insufficient number of samples (compared to the number of features) to build a model of sufficiently good quality [5]. This is the case in fields like bioinformatics, where Microarray data could contain thousands of gene expressions as features but a few tens or hundreds of samples.

To fight these challenges, effective techniques to reduce the dimensionality of datasets are necessary. A common technique for this is Feature Selection (FS). FS reduces the dimensionality of a dataset by selecting a subset of the datasets' features that are most relevant based on a defined criteria. FS methods can be classified by their reliance on a learning algorithm to select the most relevant features into *filter*, *wrapper*, and *embedded* FS methods [3].

Filter methods evaluate and select features independent of any learning algorithm by solely relying on the characteristics of the dataset to determine the relevance of a feature. This approach returns a subset of features that is not biased towards any learning algorithm and is thus highly generalizable but not optimal for any chosen learning algorithm.

Wrapper methods select features by going through an iterative process of selecting a subset of features and evaluating its quality by the performance (e.g., accuracy) of a model built using a particular learning algorithm. The cycle continues until arriving at a subset of features that optimizes the model performance or a pre-set halt condition is satisfied. Wrapper methods select a subset of features with the aim of optimizing the performance of the used learning algorithm hence lacking generalizability. Also, their iterative approach makes them time inefficient.

Embedded methods refer to learning algorithms that have FS encapsulated. Embedded methods are optimal and time-efficient because they use the target learning algorithm in the selection of features however, not iteratively. Examples of embedded methods are tree-based learning algorithms such as decision trees which prune irrelevant features while building the model, using criteria like *Entropy* or *Gini* (defined by Equations 1 and 9 respectively).

A FS method is affected by various dataset characteristics, like its number of features or instances. Therefore, it is imperative to understand the dataset characteristics that affect the performance of each FS method. Furthermore, depending on the data analysis task at hand, FS methods may impact the training time and accuracy of a model. Thus, the impact could differ for classification (binary, multiclass), clustering, and regression tasks. Besides impacting various tasks differently, their effect could differ per learning algorithm. Hence, a context-specific understanding of the impact of FS is essential.

In this work, we focus on binary and multiclass classification tasks and study the impact of eight filter FS methods on four classification algorithms. To select representative datasets from the OpenML[1] repository for the experiments, we use clustering

[1] https://www.openml.org

discretization to find two clusters each for three dataset characteristics, namely: number of features, number of instances, and class balance; while the number of classes fits by default into two clusters (binary and multiclass). This leads to 16 possible combinations of the dataset characteristics, and for each combination, we choose two representative datasets.

In particular, we make the following contributions:

(1) We identify the factors that influence the runtime of each FS method.
(2) We propose a systematic method to select representative datasets from a given repository.
(3) We perform an extensive set of experiments, with varying sizes of datasets, and show the scalability of the FS methods.
(4) We study the effect of FS on the accuracy and training time of binary and multiclass classification models and for both model types, we statistically determine the significant factors responsible for the observed effects and highlight the differences in the outcomes of both model types.
(5) We perform an extensive set of experiments with various feature subset sizes, and investigate how this impacts the observed changes in the accuracy and training time of the classifiers after FS.

2 RELATED WORK

The objective of FS is to build simpler and more understandable models, improve model performance, reduce their training time, and prepare data that are clean, understandable, and easier to visualize [17]. Due to its relevance in Big Data, researchers have focused on it, especially in recent decades. An essential aspect of this ongoing research has been: investigating the efficacy of the many proposed FS methods through benchmarks and comparisons. Many field-specific FS benchmarks have been conducted in which the datasets used are all related to a particular subject. For example, software defection data was used in [32], object-based imagery data in [16], biomedical data in [10], cancer data in [19], intrusion detection data in [21], and text categorization datasets in [2]. This re-emphasizes the usefulness of FS across domains. However, there is an intrinsic relationship between dataset characteristics and the performance of FS methods' [22]. Therefore the dataset homogeneity questions the robustness of these benchmarks as there is but little deviation in the dataset's characteristics. For example, with biomedical datasets, it is often the case that there are a large number of features and a relatively small number of instances. A benchmark using only biomedical datasets could be biased towards this factor.

The efficacy of an FS method is usually evaluated using a learning algorithms' performance. The choice of learning algorithms to use depends on the intended analysis task. By this, we mean tasks like binary or multiclass classification, regression, clustering, or any other data analysis task. For classification tasks, past benchmarks [4, 10, 22, 32] mainly focused on binary classification, generally considered easier. Some field-specific benchmarks [2, 16, 18, 21] have also focused on multiclass classification. However, there is a lack of comparative studies highlighting the distinct impact of FS on both analysis tasks.

Using learning algorithms to evaluate and compare FS methods means comparing the performance of models built with the features selected by each FS method. The most commonly used metric is the model accuracy [4, 16, 21, 22]. Some others include: ROC [2, 19], AUC [2, 10, 32], F-Score [2, 19], Precision [2], Recall [2], and Kappa index [16]. Indeed, one can compare the performance of various FS methods using any of these metrics. However, these metrics alone, do not take into account the relative impact (i.e., in comparison to the models built without FS). [4] benchmarks 23 models, 22 built using the features selected by 22 FS methods, and one built without FS. Using the mean accuracy (from 10-fold cross-validation), they compared the FS methods by counting the number of datasets (out of 16) on which the mean accuracy of each model is higher than others.

In this work, we first propose a systematic method to find representative datasets in a given repository in order to remedy the problem of 'non-representative' or 'homogeneous' datasets. Next, using this method we select 32 datasets from the OpenML[28] repository and benchmark eight filter FS methods with a relative measure, that is calculated as the change in training time (16) and accuracy (17) after FS. We use four classification algorithms in binary and multiclass classification tasks. Finally, we validate the significance of our results using statistical tests.

3 BACKGROUND

In the following, we provide the necessary definitions of the concepts appearing throughout this work.

3.1 Preliminaries

Entropy quantifies the uncertainty of a discrete random variable X with sample space $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and is defined as:

$$H(X) = -\sum_{x_i \epsilon \mathcal{X}} P(x_i) \log(P(x_i)), \tag{1}$$

where $x_i \in \mathcal{X}$ and $P(x_i)$ is the prior probability of x_i over all possible outcomes of X i.e. $\mathcal{X}$.

Conditional entropy measures the remaining uncertainty of X given another discrete variable Y with sample space $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ and is defined as:

$$H(X|Y) = -\sum_{y_i \epsilon \mathcal{Y}} P(y_j) \sum_{x_i \epsilon \mathcal{X}} P(x_i|y_j) \log(P(x_i|y_j)), \tag{2}$$

where $P(x_i|y_j)$ is the conditional probability of x_i given y_j.

Mutual information is the measure of information shared between two discrete random variables X, Y and is defined as:

$$I(X;Y) = H(X) - H(X|Y). \tag{3}$$

Conditional mutual information quantifies the amount of information shared between two discrete random variables X, Y when a third discrete random variable Z is known. It is defined as:

$$I(X;Y|Z) = H(X|Z) - H(X|Z,Y). \tag{4}$$

Bayes' theorem states that given two events A and B, with $P(B) \neq 0$, then:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{5}$$

3.2 Classification Algorithms

The classification task is a common problem in practice and is categorized into (i) binary classification where subjects are assigned to one of two classes; for instance the bank wants to know if a loan applicant is likely to default or not before issuing a loan and, (ii) multiclass classification where subjects are assigned to one of more than two classes, as in classifying the image of a plant into one of many species.

In this work, four classification algorithms that do not have FS embedded, and that are applicable to both binary and multiclass problems were used. Their details are discussed below.

Naive Bayes (NB) is a collection of classification algorithms that are based on Bayes' theorem and the assumption that every pair of features F_i, F_j in a dataset are conditionally independent given the class feature Y i.e.:

$$P(F_i \cap F_j|Y) = P(F_i|Y)P(F_j|Y). \tag{6}$$

NB classifies an instance to the most probable class value y given its feature vector $\langle f_1, f_2, \ldots, f_m \rangle$ and has a time complexity of $O(m.n)$ [11] for a dataset with m features and n instances.

K-Nearest Neighbours (KNN) is a commonly used classification algorithm that works on the 'bird of the same feather flock together' principle. The class of an unknown instance is determined by that of its K closest neighbors whose classes are known. KNN is considered a lazy algorithm because it does not build a classifier once for reuse, rather it scans through the training data to determine the class of an unknown instance each time it is called [20], and has a time complexity of $O(m.n)$ [9].

Linear Discriminant Analysis (LDA) is a classic classification algorithm that uses linear decision boundary(ies) to build a classifier for unknown instances. Its closed-form solution[2] with no need for hyperparameter tuning makes it easier to compute and it has demonstrated good performance in practice [23]. Given an unknown instance x, the probability of x belonging to a class y of the target feature Y with c classes is estimated by:

$$P(Y = y|X = x) = \frac{P(y)P(x|y)}{\sum_{l=1}^{c} P(l)P(x|l)}. \tag{7}$$

However, $p(x|y)$ can be estimated with a Gaussian distribution function which when substituted leads to a simplified discriminant function for class y given instance x defined as:

$$D_y(x) = x\frac{\mu_y}{\sigma^2} - \frac{\mu_y^2}{2\sigma^2} + \ln(P(y)), \tag{8}$$

where μ_y is the mean value of x for the class y and σ^2 is the variance across all inputs x [6]. LDA works under the assumption that the data is normally distributed and that the variance of all features are equal. It can also be used for dimensionality reduction by projecting the training data into a linear subspace such that the separability between classes is maximized [23]. LDA has a cubic time complexity of $O(m.n.t + t^3)$ [7], where t is the $\min(m, n)$.

Multilayer Perceptron (MLP) is a type of feedforward artificial neural network consisting of three layers of nodes; an input layer, output layer[3], and an arbitrary number of hidden layers. A full pass of data (epoch) through the MLP comprises forward and backward passes. In the forward pass, data is passed from the input layer through the hidden layer(s) to the output layer adjusting the weights of the nodes based on the ground truth. The backward pass goes in the opposite direction from the output layer to the input layers to minimize the error of the MLP. The classification model is built by several epochs of the training data through the MLP after which it is used to classify unknown instances.

The time complexity of MLP, $O(n.m.h^k.o.i)$ [1], depends on its architecture and configuration, i.e. the number of iterations i, output neurons o, and hidden layers k each with h neurons. MLP is applied in various fields including speech and image recognition, and natural language processing. [30] This classifier has the option of regularization which can be considered embedded FS, however this option was disabled for this work.

[2] A closed-form solution (expression) is any formula that can be evaluated in a finite number of standard operations.

[3] The number of classes in the target feature is the number of nodes in the output layer.

3.3 Feature Selection Algorithms

The generalizability of filter methods allow for easier comparison of their impact on multiple learning algorithms since the resulting subset of features is derived independently of any learning algorithm (unlike wrapper and embedded FS methods). Also, filter methods are less time-consuming than wrapper methods because they do not require an iterative training of the learning algorithm to select a subset of dataset features. For these reasons, eight filter methods were studied in this work and presented below.

Gini index is a very common statistical measure used to quantify the ability of a feature to separate instances between classes [17]. A feature F_k with r distinct values is scored as:

$$\begin{aligned} J_{Gini}(F_k) = \min_{W_{F_k}} \Big(& p(W_{F_k})(1 - \sum_{y=1}^{c} p(C_y|W_{F_k})^2) \\ & + p(\overline{W}_{F_k})(1 - \sum_{y=1}^{c} p(C_y|\overline{W}_{F_k})^2) \Big), \end{aligned} \tag{9}$$

where C_y is the $y_{th} (\leq c\ total\ classes)$ class, and W_{F_k} and $\overline{W}_{F_k}$ denotes instances with F_k value $\leq \phi$ and $> \phi$ respectively where ϕ is some j^{th} value of F_k, this implies that ϕ separates the dataset into W_{F_k} and $\overline{W_{F_k}}$. The maximum gini score in binary classification is 0.5 and lower values imply higher relevance.

ReliefF is a similarity based method which extends the classic relief method, not just for binary, but also multi-class classification [15]. The idea of this method is to select features that maximally distinguish between instances that are near to each other [25]. The score of a feature F_k is defined as:

$$\begin{aligned} J_{ReliefF}(F_k) = \frac{1}{c} \sum_{j=1}^{l} (& -\frac{1}{m_j} \sum_{x_r \epsilon NH(j)} d(X(j,k) - X(r,k)) \\ & + \sum_{y \neq y_j} \frac{1}{h_{jy}} \frac{P(y)}{1 - P(y)} \sum_{x_r \epsilon NM(j,y)} d(X(j,k) - X(r,k))), \end{aligned} \tag{10}$$

where l is the number of randomly selected instances ($\leq n$), $NH(j)$ of size m_j is the set of instances closest to instance x_j and in the same class, $NM(j, y)$ with size h_{jy} is the set of instances closest to x_j and in class y, $P(y)$ is the probability of instances belonging to class y, and $d(.)$ is the distance estimator between two instances.

Spectral Feature Selection (SPEC) is a similarity-based method that works for supervised and unsupervised tasks [17]. SPEC selects features from a dataset X by the following three steps [33]: (i) it builds a similarity set S from X using a similarity measure, e.g., Radial Basis Function (RBF) kernel, (ii) it extracts a representative graph G from S, and (iii) lastly, the structural information of G obtained from its spectrum[4] determines the relevance of each feature.

Conditional Mutual Information Maximization (CMIM) is an information theoretical-based method which iteratively picks features that maximize mutual information to the class feature conditioned on the already selected features [29]. Where S is the subset of already selected features, Y the class feature, and

[4] The spectrum of a graph refers to its eigenvalues and their multiplicities.

F_k an unselected feature, the feature score of F_k can be defined as:

$$J_{CMIM}(F_k) = \min_{F_j \epsilon S} I(Y; F_k|F_j). \quad (11)$$

This approach ensures that the selected features are good predictors of the class label but minimally redundant.

Minimum Redundancy Maximum Relevance (mRMR) is also an information theoretical-based method proposed to select features that balance relevance and redundancy. A feature F_k is scored as:

$$J_{MRMR}(F_k) = I(Y; F_k) - \frac{1}{|S|} \sum_{F_j \epsilon S} I(F_k; F_j), \quad (12)$$

where the mutual information between the feature and target feature $I(Y; F_k)$ is the measure of relevance while $\frac{1}{|S|} \sum_{F_j \epsilon S} I(F_k; F_j)$ tells the features redundancy compared to already selected features [4].

Joint Mutual Information (JMI) is an information theoretical-based method which iteratively selects features that complement already selected features with respect to the class feature. Its score is defined as:

$$J_{JMI}(F_k) = \sum_{F_j \epsilon S} I(Y; F_k|F_j). \quad (13)$$

Efficient and Robust Feature Selection (RFS) is a sparse learning based method that uses a joint $l_{2,1}$-norm minimization on both the loss function and the regularization. This approach is more robust to noise and achieves group feature sparsity [17]. The objective function for RFS is:

$$\min_W ||XW - Y||_{2,1} + \alpha||W||_{2,1}, \quad (14)$$

where W, X, and Y are the matrix of feature weights, matrix of feature values and one-hot encoding of the target feature, respectively and α is a parameter of the regularization term $||W||_{2,1}$.

Correlation-based Feature Selection (CFS) is a statistical based method with the basic idea of selecting a feature subset S in which features are highly correlated with the class and uncorrelated with other selected features [13]. This implies that redundant features are less likely to be selected. The CFS score of a feature subset dubbed "merit" is defined as:..

$$J_{CFS}(S) = \frac{|S|\overline{r_{cf}}}{\sqrt{|S| + |S|(|S| - 1)\overline{r_{ff}}}}, \quad (15)$$

where $\overline{r_{cf}}$ is the average feature-class correlation and $\overline{r_{ff}}$ is the average feature-feature inter-correlation.

CFS begins with an empty set of features, computes the symmetric uncertainty of each feature and then greedily adds features to the set using forward best-search heuristic strategy [13]. It stops after five consecutive searches with improvement in merit .

4 METHODOLOGY

To evaluate the impact of the presented FS methods on classification, we followed a systematic approach to define the components of the experiments carried out in this work and to select the datasets used. In what follows, we present the details.

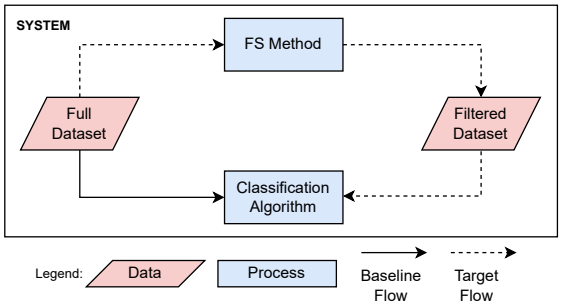


Figure 1: System definition showing components and flow.

4.1 Performance Evaluation

The number of features to select, parameters of FS methods and classification algorithms, and dataset characteristics culminate into more factors than we can control in this work. Therefore, it is essential to clearly define the components of the experiments, as failing to do so can lead to mistakes (such as unrepresentative workload, erroneous analysis, or too complex analysis). We followed the systematic approach for performance evaluation proposed in [14] to avoid these common mistakes.

In what follows, we present the details of the ten steps in the systematic approach for performance evaluation.

(1) **Goals and System Definition**
In this work, the system under performance evaluation consists of the datasets (full and filtered), FS methods, and classification algorithms as shown in Figure 1. Within this system, we define two subsystems: (i) the **target** which represents the flow ⟨*Full Dataset* → *FS Method* → *Filtered Dataset* → *Classification Algorithm*⟩, where FS is performed before model building and (ii) the **baseline** which represents the path ⟨*Full Dataset* → *Classification Algorithms*⟩ where the full dataset, without FS, is used to build the model.
The experiments we execute benchmark the target against the baseline subsystem for all the combinations of dataset, FS method, and classification algorithm in this work.
The goals of the experiments conducted are to:
- Measure the FS runtime for each dataset.
- Measure the model training time of the classifiers built in the baseline and target subsystems.
- Measure the performance of the classifiers built in the baseline and target subsystems.

(2) **Services and Outcomes**
The FS methods and learning algorithms deliver the services of the system. Both processes take a dataset as input; the former outputs a subset of the datasets' features while the latter produces a classifier model.

(3) **Metrics**
Metrics are the criteria used to evaluate the performance of a system. The metrics used to evaluate the system under performance evaluation (Figure 1) are:
- **FS runtime**, which is the time taken to obtain the feature subset/ranked features of a dataset.
- **Model Training Time (TT) change** is the ratio of the difference in time between training the baseline and target classifiers to the baseline classifier training time.

It is expressed as:

$$TT\ change^5 = \frac{Baseline\ TT - Target\ TT}{Baseline\ TT}. \quad (16)$$

- **Classifier accuracy change** is the ratio of the difference between the balanced accuracy of the target classifier and the baseline classifier to the balanced accuracy of the baseline classifier expressed as:

$$Accuracy\ change^6 = \frac{Target\ Accuracy - Baseline\ Accuracy}{Base\ Accuracy}. \quad (17)$$

(4) **Parameters**
Several characteristics of the system components (datasets and algorithms) affect the performance of the system. These are called parameters and are categorized into workload and system parameters. The workload parameters are associated with the dataset and algorithms and include:
Dataset parameters
- Number of features (m)
- Number of instances (n)
- Number of classes (c)
- Dimensionality, which is the feature-instance ratio of a dataset defined as:

$$Dimensionality = \frac{m}{n}. \quad (18)$$

- Class Balance (CB), this is the distribution of instances amongst the classes in the target variable. It is measured by the Shannon entropy which is defined as:

$$CB = -\frac{\sum_{i=1}^{c} \frac{s_i}{n} \log(\frac{s_i}{n})}{\log c}, \quad (19)$$

where there are s_i instances in class i. A perfectly balanced dataset with equal number of instances in each class has a class balance of one while an extemely unbalanced dataset has a class balance of zero.
- Class Entropy, this measures the entropy (Equation 1) of the classes in the target feature.
- Average feature correlation, this measures the interdependence between all features and is quantified by

$$p = \frac{1}{T} \sum_{i=1}^{c} \sum_{j=1}^{m-1} \sum_{l=j+1}^{m} |P_{jl}|, \quad (20)$$

where P_{jl} is the Pearson's correlation coefficient of features j and l, and T is the total number of P_{jl}'s summed. A p value of zero indicates independence of all features while a value of one indicates high correlation between features and thus feature redundancy [27].

Algorithm parameters
- Feature subset size (k), which is the number of features to be selected; affects the results derived from the system.
- Hyper-parameters, this refers to the several parameters in FS and classification algorithms that can be set to differing values to control the model building process [31]. These parameters affect the output of the algorithms and thus the system.

Lastly, the **system parameters** are the characteristics of the hardware used to execute the experiments for the system performance evaluation and include among others:
- Processor speed
- RAM/Disk size
- Operating system context switching overhead

(5) **Factors to Study**
The number of parameters controlled in an experiment determines its workload and completion time. Considering the exponential growth of the experimental design space with an increasing number of parameters, we considered the five parameters: *number of features, instances, and classes, class balance, and feature subset size*, as more important because they control the time complexity of the FS methods as shown in Table 4, as well as classification accuracy [22]. Also, some other workload parameters are derived from the selected factors or are known to not influence the results of FS methods, for instance, dimensionality is simply the feature-instance ratio and most FS methods do not factor in the average correlation of features during FS, but rather assess features individually. Hence, *dimensionality* and *average feature correlation* are not included in the study.
Factors such as processor speed or storage size, associated with the hardware on which the experiments were executed were constant as one machine was used for all executions. However, to mitigate the effects of context switches, especially for execution time, we employed 10 fold cross-validation to ensure accurate results. Lastly, the default hyper-parameters of the classification and FS algorithms were used since the same setting was applied to both baseline and target classifiers. Controlling all parameters for all algorithms would not only explode the design space but also shift our focus to hyper-parameter optimization, which is outside the scope of this work.

(6) **Evaluation Technique**
The experiments were programmed with Python, which has the scikit-feature[7] repository, scikit-learn[8], and time[9] libraries that implement the studied FS and learning algorithms as well as allow us to measure execution times.

(7) **Workload**
The datasets used in the evaluation of the system constitute its workload. Considering four datasets characteristics with two clusters each (see Section 4.2), we systematically selected 32 real-world datasets from the OpenML repository for this work. We considered only datasets with no missing values as candidates to save time in the preprocessing stage and avoid the potential effect of missing values on the classifier's performance (especially in cases with missing values being dropped. Lastly, datasets with only numeric values were considered due to the input constraint by the FS methods studied.

(8) **Experiment Design**
Out of the eight FS methods studied, Gini, RFS, ReliefF, SPEC, and CFS methods return a ranking of the features in order of relevance, while JMI, MRMR, and CMIM return a subset of the features according to the specified size. Five possible subset sizes were studied, specifically $(number\ of\ features)^i$, for i ∈ [0.5, 0.6, 0.7, 0.8, 0.9]. This means that a total of 20 runs (Table 1) were required for the eight FS methods and two classifiers (baseline and target) for each of the four classification algorithms. Therefore

[5]There is an expected decrease in runtime i.e., Baseline TT > Target TT.
[6]There is an expected increase in accuracy i.e., Target Accuracy > Baseline Accuracy.

[7]https://github.com/jundongl/scikit-feature
[8]https://scikit-learn.org/stable
[9]https://docs.python.org/3

Table 1: Required number of runs for each feature selection method.

Feature selection Method	Required runs
Gini, RFS, ReliefF, SPEC,CFS	1
JMI, MRMR, CMIM	5
Total	**20**

with all 32 datasets, a full factorial experimental design with 5,120 (32*20*8) experiments will be used.

(9) **Data Analysis and Interpretation**
To derive insights from the results of the experiments, Z-test, Friedman, and Nemenyi statistical methods were used to analyze our results (i.e., measured metrics). For each of these tests, a null and alternative hypothesis were formulated and based on the test result, we either reject or fail to reject the set null hypothesis.

(10) **Results**
Lastly, conclusions from the analysis of the results of the experiments are presented and discussed in Section 5.

4.2 Data Selection

Several data repositories provide open data which enables research; OpenML was used in this work. Considering the input constraint of some of the algorithms studied, not all datasets in OpenML (over 21,000) could be used.
Available datasets were pruned based on the following eligibility criteria:

- Number of features $\leq$ 1.000
- Number of instances $\leq$ 10.000
- No missing values
- Numerical types (excluding timestamps)
- Single target variable

After pruning, there were 380 candidate datasets available and the metadata (number of features, instances, classes, and the class balance) of these datasets showed a wide range of values for the factors to be studied; for example, the number of features of the datasets ranged from 2 to 971. Also, we noticed repeated or highly identical datasets. So, we selected a subset that properly represents the available datasets while reducing redundancy.

Using the clustering discretization method which takes into account the intrinsic distribution of data values, each of the factors in the metadata were discretized into two bins. This was done for both the metadata values and the logarithm of the values as shown in Tables 2 and 3 respectively. It is indeed possible to discretize into more than two bins, however, this will lead to an exponential increase in the full factorial of the design space. With two bins and 4 dataset factors, we had 2^4 possible factor combinations all of which need representative datasets. However, with three bins, only nine out of 81 combinations had representative datasets. Increasing the number of bins will therefore result in more factor combinations with no representative datasets.

The discretization results from applying clustering on the log values was used to guide the selection of the datasets from OpenML because it best describes the distribution of the datasets in the repository and contained candidate datasets for all 16 combinations of the studied factors.

After the discretization, two datasets were selected randomly for each of the 16 factor combinations with a total of 32 datasets.

Table 2: Discretization of values of studied factors using the clustering method.

Factor	Bin	Number of datasets
Number of instances	0: [27, 2.600]	320
	1: [3.107, 9.989]	60
Number of features	0: [2, 618]	373
	1: [785, 971]	7
Class balance	0: [0,0385, 0,8083]	84
	1: [0,8232, 1,0]	296

Table 3: Discretization of log values of studied factors using the clustering method.

Factor	Bin	Number of datasets
Number of instances	0: [27, 846]	246
	1: [937, 9.989]	134
Number of features	0: [2, 16]	205
	1: [19, 971]	175
Class balance	0: [0,0385, 0,8083]	84
	1: [0,8232, 1,0]	296

4.3 Execution

To begin, all features of the dataset were discretized using the uniformed discretization method; setting the number of bins to $\max\{\min\{\frac{n}{3}, 10\}, 2\}$ where n is the number of instances in the dataset [4]. Discretizing the features is done to meet the input constraint of most FS methods which work with only discrete features.
Next, the dataset was split into ten partitions; nine partitions formed the training set, while one partition was used as the test set. After this split, the entire training set was used for FS as well as building the base classifier while the training data containing only the selected subset of features were used to build the target classifier. Both classifiers were tested on the test set and the accuracy was measured. The FS, model building, and testing are repeated ten times for each dataset (i.e., 10-fold cross-validation); each time a new partition is used as the test data while the remaining nine are used as the training data. Figure 2 gives an overview of the described execution, from discretization to measure recording. At the end of each of the 10-fold execution, the median runtime and average balanced accuracy were recorded. The median was used in the runtime for robustness against possible variations due to context switching, and since we use a different training set for each fold, there might be slight differences in the accuracy, hence the average serves as a global accuracy.

Due to the poor stability of some FS methods, different features might be selected with a change of data samples. Hence, FS and model building were executed together for each iteration to ensure that the performance evaluation is done on the same data. The Python source code and datasets information for the experiments are publicly available on Github[10].

5 RESULTS

There is an expectation for improved model runtime and accuracy after FS. However, FS itself could be time expensive leading to an overhead that causes the total time needed for FS and model

[10]https://github.com/F-U-Njoku/filter-fs-impact-on-classification

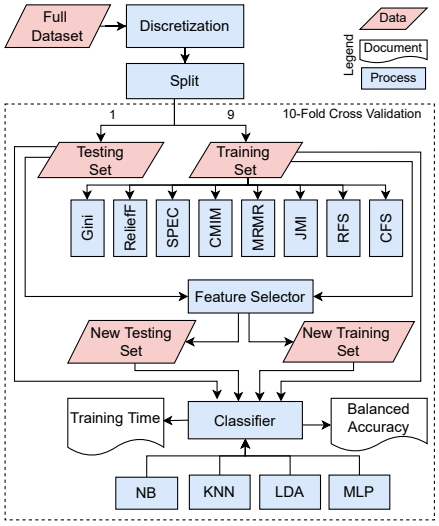


Figure 2: Experiment execution flow, from data ingestion to metric result collection.

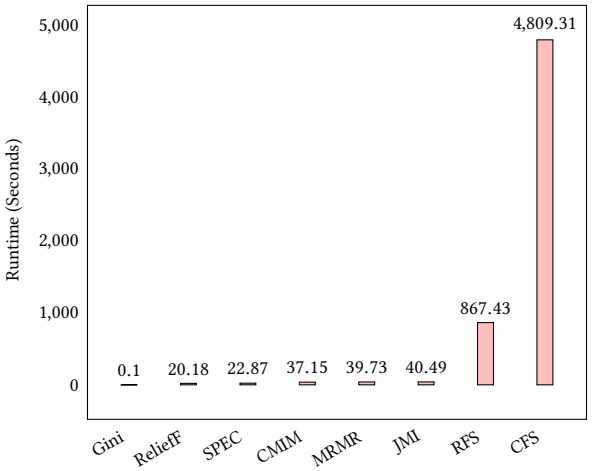


Figure 3: Average runtime for FS by each method on all datasets.

building to increase beyond the time it takes to build the model without FS. In this section, we present and discuss the results of our experiments on eight FS methods and four classification algorithms using 32 datasets with feature subset size set to $m^{0.5}$, $m^{0.6}$, $m^{0.7}$, $m^{0.8}$, and $m^{0.9}$ where m is the number of features.

5.1 FS Runtime

The runtime of a FS method is affected by various factors and in differing proportions. Some are influenced more by the number of features, others by the number of instances, and so on. The time complexity of the studied FS methods are presented in Table 4, this shows the factors which affect the runtime of each method.

Gini has the lowest runtime and is influenced mainly by the feature and class size. ReliefF and SPEC are controlled by the instance and feature size, with the former having a quadratic impact on the runtime. CMIM, MRMR, and JMI all have the same time complexity, with feature and subset size being the factors that predominantly influence their runtime. Lastly, RFS and CFS have the longest runtimes due to the numerous matrix multiplication operations required.

Table 4: Time complexity of FS methods.

Algorithm	Time complexity	Source
Gini	$O(mc)$	[17]
ReliefF	$O(n^2m)$	[26]
SPEC	$O(n^2m)$	[33]
CMIM	$O(mk)$	[12]
MRMR	$O(mk)$	[24]
JMI	$O(mk)$	[17]
RFS	$O(t(2m^2n + (m+c)n^2 + mnc + m))$	[17]
CFS	$O((n+1)((m^2-m)/2)+$ $(m^2-m)/2 + k + (k^2-k)/2$	[13]

Figure 3 shows the recorded average runtime for FS on the 32 datasets for each FS method spanning from less than a second to over 80 minutes. Of the eight methods, CFS had the longest runtime for feature selection taking an average of 80 minutes for all datasets; this is due to the multiple computations of pairwise correlations between features required by this method. Second to CFS was RFS which took an average of 14 minutes. The other methods took less than 50 seconds on average, while Gini was the most efficient, taking less than a second on average.

Although FS is time expensive for some methods, the advantage of filter methods is their generalizability. The selected features can be used repeatedly across various learning algorithms for model selection since they were obtained independently of any learning algorithm.

5.2 Model Training Time

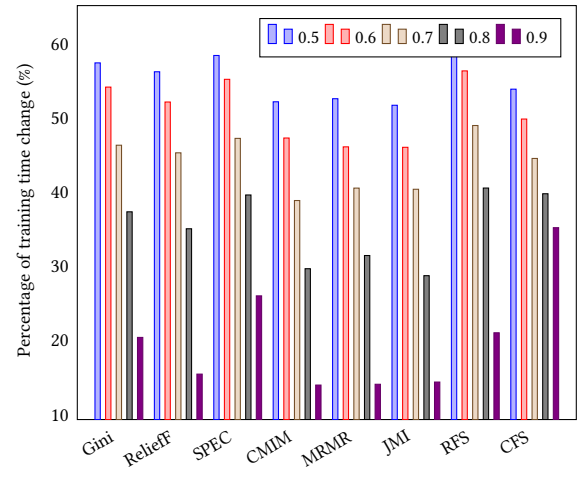
The dataset size, particularly the feature and instance sizes, control the time it takes to build a classifier, as discussed in Section 3. For learning algorithms that build classifiers after training, such as NB, LDA, and MLP, FS reduces the model training runtime as expected. For LDA, we recorded on average for all datasets (i.e., both binary and multiclass classification), over 50% training time reduction when the feature subset size, k was set to $m^{0.5}$ and over 13% for k set to $m^{0.9}$ as shown in Figure 4a. In this case, as expected, selecting fewer features led to faster model building.

On the other hand, a lazy learning algorithm like KNN does not build a classifier at the point of training; rather, it stores useful metadata that is used at the point of classification. Hence, KNN does not benefit from FS in the expected manner. As shown in Figure 4b, KNN follows a pattern different from other algorithms and gives the best training time improvement with k set to $m^{0.8}$ in most cases. Therefore, the benefit of reduced model runtime after FS depends on the learning algorithm.

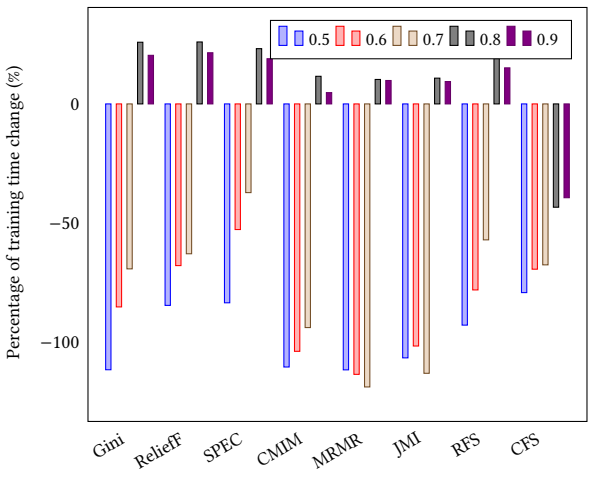
5.3 Accuracy

Accuracy is one of the methods used to measure the performance of a model and due to the presence of unbalanced datasets in the workload, we specifically use the balanced accuracy.

Since one of the objectives for FS is to build models with improved performance that generalize the data better, the expectation is to have an improvement in the model accuracy after FS. However, as shown in Figure 6, the change in accuracy after FS differs for binary and multiclass classifications. For binary classification, there was an average improvement of the accuracy after FS for all methods except SPEC. In contrast, reducing the number of features generally led to accuracy decrease for multiclass



(a) LDA training time change.



(b) KNN training time change.

Figure 4: Average percentage change in the training time of LDA and KNN classifers after applying each FS method to all datasets (i.e., binary and multiclass classification), with varying subset sizes. Positive and negative percentages imply a reduction and an increase in the training time, respectively, compared to the baseline training time.

classification. Although SPEC is applicable to both supervised and unsupervised learning tasks, it gave the worst performance for both binary and multiclass classification as shown in Figures 5 and 6; this is because it performs FS independent of the target variable; hence, the selected features by SPEC tend to be less relevant compared to those selected by other methods.

Also shown in Figure 5, CFS yielded good results in most cases; this is because it uses the correlation of features to the target feature as well as the pairwise correlation of features for FS. This is useful because it ensures minimal redundancy in the selected features. For multiclass classification, CFS and MRMR gave the best result (i.e. the least accuracy degradation). Therefore, although FS generally results in a decrease in multiclass classification accuracy, in the case that the dataset is too large and FS is indeed needed, CFS and MRMR methods are suggested.

Since the improvement in accuracy is different for binary and multiclass classifications, we analyze the change in accuracy for both classification types separately with the aim of identifying the factors that most influence the accuracy change.

5.3.1 *Statistical Analysis.*

By visualization, we appreciate a difference in the accuracy change obtained for binary and multiclass classifications (Figure 5) and this is irrespective of the FS method used (Figure 6). We further our investigation by testing the significance of this observation statistically for the subset size set to $m^{0.7}$ and an alpha level of 0.05.

We begin by making the following hypothesis:

- H_0: The classification type does not influence the change in accuracy derived from applying FS.
- H_1: The classification type influences the change in accuracy derived from applying FS.

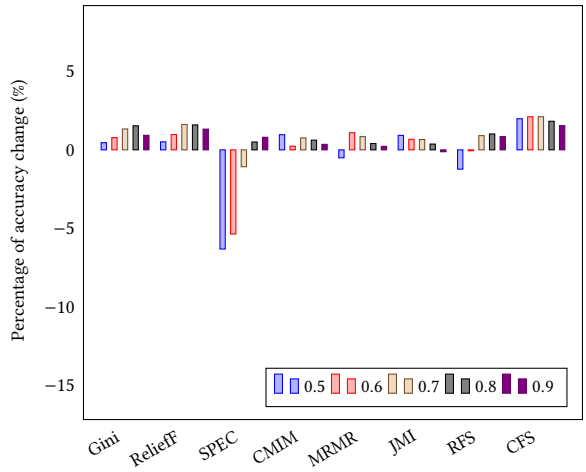
To test the aforementioned hypothesis, we consider two independent groups. The binary and multiclass groups, each made up of accuracy change between baseline and target classifiers after FS on the 16 datasets (512 result points). Testing the previous hypotheses using Z-test, the results show the impact of FS on binary classification in terms of accuracy change ($\mu = 0.0058, \sigma = 0.0506, N = 508$) was significantly higher than on multiclass classification ($\mu = -0.0627, \sigma = 0.1483, N = 512$), $Z = 9.2589$, $p < 0.001$. Therefore, the classification type indeed influences the change in accuracy derived from applying FS.

Besides the type of classification, the various classification algorithms could interact with the FS methods to give different results in terms of accuracy change. Hence, using Freidmans' test, we evaluate the accuracy changes for binary and multiclass classification separately. Friedman (F_f) test [8] ranks the accuracy change of the four classifiers for each *dataset-FS method* pair with the highest rank being 1 and the smallest rank 4. In the case of ties, the average rank is used. We test the null hypothesis that all algorithms benefit equally from FS. Therefore there should be an equal ranking of the derived accuracy change for each dataset and FS selection pair.

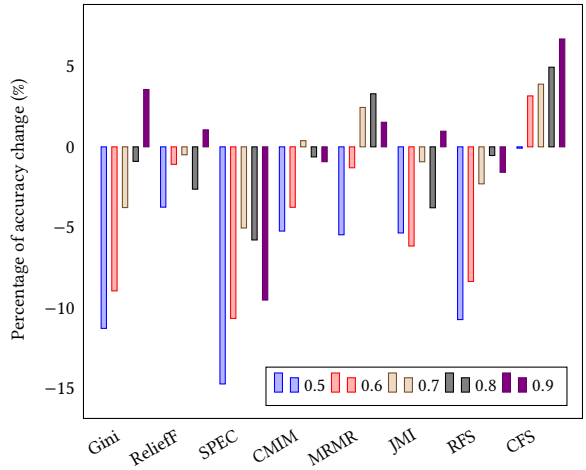
For binary classification, using 16 datasets[11], eight FS methods, and four classification methods, the F_f-statistic is distributed according to the F distribution with $(4-1)$ and $(4-1)(127-1) = 378$ degrees of freedom. The critical value of F(3,378) is 2.6285, and the derived F-statistic is 2.47405553 giving $p = 0.0612$. Therefore, we fail to reject the null hypothesis and conclude that the studied classification algorithms similarly gain from FS for binary classification.

For multiclass classification, using 16 datasets, eight FS methods, and four classification methods, the F_f-statistic is distributed according to the F distribution with $(4-1)$ and $(4-1)(128-1) = 381$ degrees of freedom. The critical value of F(3,381) is 2.6283 and the derived F-statistic is 11.41980269 giving $p < 0.001$. Therefore we reject the null hypothesis and proceed to do a post-hoc test. We use the Nemenyi test with critical value 2.569 to compare the classifiers to each other, which yields a Critical Difference (CD) of 0.4145. Using the CD, we identify two groups of algorithms as shown in Figure 7: LDA and MLP benefit more in terms of accuracy improvement after FS compared to KNN and NB. Therefore, for multiclass classification, the impact of FS on accuracy is affected by the choice of the algorithm.

[11]For CFS FS method, the largest dataset was left out as it was not completed within reasonable time.



(a) Binary accuracy change.



(b) Multiclass accuracy change.

Figure 5: Average percentage change in the accuracy of the LDA classifer after applying each FS method to all datasets, with varying subset sizes.

5.4 Recommendations

From the results of our experiments presented previously and in the supplementary material[12], it is clear that there is no FS method that is best for all cases, i.e., all datasets, tasks, and learning algorithms. Choosing a feature selection method is, therefore, use-case dependent. The first step is to define the analysis task for which we considered binary and multiclass classification. Feature selection behaves differently for these tasks, particularly in terms of how it impacts model performance and execution time.

For multiclass classification, feature selection does not improve the model performance (i.e., accuracy); on the contrary, it degrades it; and the smaller the feature subset size, the worse the decline in the performance. For this reason, our analysis suggests that feature selection should be avoided for multiclass classification. Yet, in case necessary (e.g., for performance/training time reasons), then either CFS (for small-sized datasets) or MRMR (for large-sized datasets) can be used, because they gave the least

[12]https://github.com/F-U-Njoku/filter-fs-impact-on-classification

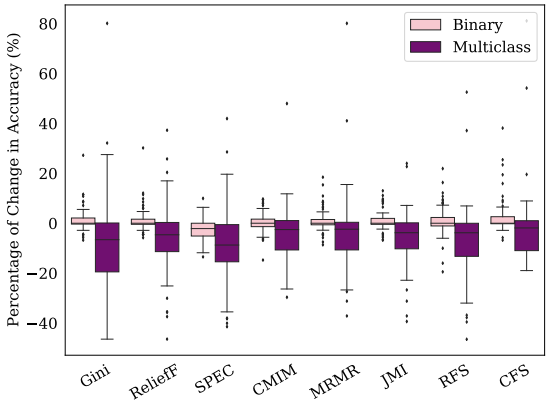


Figure 6: Percentage change in the classification accuracy of the classifiers built before and after FS on all 32 datasets for each FS method.

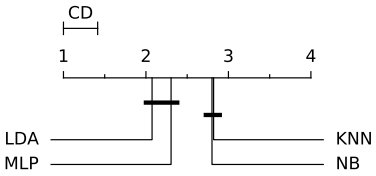


Figure 7: Comparison of all four classifiers (multiclass) against each other using Nemenyi test. Groups of classifiers that are not significantly different i.e.classifiers that benefit similarly from FS are connected.

accuracy degradation and improvement in a few cases compared to the other studied methods.

In the case of binary classification, we observed improvements in the classification accuracy, which depends on the chosen classification algorithm. For all five feature subset sizes that we set, CFS and MRMR increased the model accuracy for the NB classifier. For the KNN classifier, CFS, ReliefF, Gini, and CMIM improved accuracy in that order. For the LDA classifier, CFS, ReliefF, Gini, and CMIM improved accuracy in that order , and lastly, the MLP classifier with CFS led to improvement in accuracy. It is tempting to assume that CFS is the best choice to make when the goal is to improve performance for binary classification. However, we must recall from Figure 3 that this method is computationally expensive and is unsuitable for large datasets. Hence these recommendations must be considered taking into account the dataset size and FS methods complexity as presented in 5.1.

Finally, whether binary or multiclass classification, Gini is a time-efficient method, and especially for very large datasets, we propose this feature selection method to improve runtime. Figure 8, summarizes our recommendations based on the results of our experiments and given the goal, task, and classification algorithm.

6 CONCLUSIONS AND FUTURE WORK

In this work, we studied the impact of eight filter FS methods on four classification algorithms using 32 real-world datasets. We

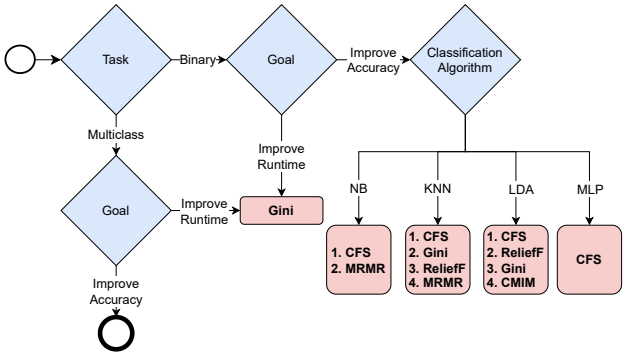


Figure 8: Proposed guidelines for choosing feature selection methods with aim of runtime or accuracy improvement.

found that on average, FS improves the accuracy for binary classification, however, for multiclass classification, feature selection leads to a decline in model accuracy.

With regards to FS runtime, the Gini FS method was the most efficient with an average runtime of less than a second making it the recommended FS method when minimizing runtime is the objective. However, when the goal is to improve classifier accuracy, none of the studied FS methods gives best accuracy improvement in all cases. Although FS on multiclass classification on the average led to a degradation in classifier accuracy, we observed that CFS and MRMR methods gave the best result in that they yielded the least degradation in accuracy and led to improvement in some cases.

As further work, we plan to extend this work to larger datasets with more clusters of dataset factors using multiple repositories to facilitate finding representative datasets. Also, we intend to investigate the dependence of multiclass classification performance degradation after FS on the multiclass classification strategy and the KNN training time degradation with the reduction in the number of features for some cases.

Since FS applies to supervised and unsupervised tasks, this work can be extended to regression and clustering tasks. Lastly, several existing methods have polynomial runtimes and do not scale efficiently; more efficient implementations of these methods aiming for linear or sub-linear time complexity are highly needed, especially with Big Data being ubiquitous..

7 ACKNOWLEDGMENTS

The project leading to this publication has received funding from the European Commission under the European Union's Horizon 2020 research and innovation programme (grant agreement No 955895).

REFERENCES

[1] Hajar Alla, Lahcen Moumoun, and Youssef Balouki. 2021. A Multilayer Perceptron Neural Network with Selective-Data Training for Flight Arrival Delay Prediction. *Scientific Programming* 2021 (2021).

[2] Yindalon Aphinyanaphongs, Lawrence D Fu, Zhiguo Li, Eric R Peskin, Efstratios Efstathiadis, Constantin F Aliferis, and Alexander Statnikov. 2014. A comprehensive empirical comparison of modern supervised classification and feature selection methods for text categorization. *Journal of the Association for Information Science and Technology* 65, 10 (2014).

[3] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. 2013. A review of feature selection methods on synthetic data. *Knowledge and information systems* 34, 3 (2013).

[4] Andrea Bommert, Xudong Sun, Bernd Bischl, Jörg Rahnenführer, and Michel Lang. 2020. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis* 143 (2020).

[5] Gianluca Bontempi. 2021. *Statistical foundations of machine learning (2nd edition) handbook.*

[6] Jason Brownlee. 2016. *Master Machine Learning Algorithms: discover how they work and implement them from scratch.* Machine Learning Mastery.

[7] Deng Cai, Xiaofei He, and Jiawei Han. 2008. Training linear discriminant analysis in linear time. IEEE.

[8] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006).

[9] Zhenyun Deng, Xiaoshu Zhu, Debo Cheng, Ming Zong, and Shichao Zhang. 2016. Efficient kNN classification algorithm for big data. *Neurocomputing* 195 (2016).

[10] Peter Drotár, Juraj Gazda, and Zdenek Smékal. 2015. An experimental comparison of feature selection methods on two-class biomedical datasets. *Computers in biology and medicine* 66 (2015).

[11] Charles Elkan. 1998. Naive Bayesian Learning. (12 1998).

[12] François Fleuret. 2004. Fast binary feature selection with conditional mutual information. *Journal of Machine learning research* 5, 9 (2004).

[13] Mark Andrew Hall. 1999. Correlation-based feature selection for machine learning. (1999).

[14] Raj Jain. 1990. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.* John Wiley & Sons.

[15] Igor Kononenko. 1994. Estimating attributes: Analysis and extensions of RELIEF. In *European conference on machine learning*. Springer.

[16] Andrea S Laliberte, DM Browning, and Albert Rango. 2012. A comparison of three feature selection methods for object-based classification of sub-decimeter resolution UltraCam-L imagery. *International Journal of Applied Earth Observation and Geoinformation* 15 (2012).

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

[18] Hsi-Che Liu, Pei-Chen Peng, Tzung-Chien Hsieh, Ting-Chi Yeh, Chih-Jen Lin, Chien-Yu Chen, Jen-Yin Hou, Lee-Yung Shih, and Der-Cherng Liang. 2013. Comparison of Feature Selection Methods for Cross-Laboratory Microarray Analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 10, 3 (2013).

[19] Munirah Mohd Yusof, Rozlini Mohamed, and Noorhaniza Wahid. 2016. Benchmark of feature selection techniques with machine learning algorithms for cancer datasets. In *Proceedings of the ICAIR-CACRE.*

[20] Antonio Mucherino, Petraq J. Papajorgji, and Panos M. Pardalos. 2009. *k-Nearest Neighbor Classification.* Springer New York.

[21] Hai Thanh Nguyen, Slobodan Petrović, and Katrin Franke. 2010. A comparison of feature-selection methods for intrusion detection. In *International conference on mathematical methods, models, and architectures for computer network security*. Springer, 242–255.

[22] Dijana Oreski, Stjepan Oreski, and Bozidar Klicek. 2017. Effects of dataset characteristics on the performance of feature selection techniques. *Applied Soft Computing* 52 (2017).

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

[24] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Transactions on pattern analysis and machine intelligence* 27, 8 (2005).

[25] Marko Robnik-Šikonja and Igor Kononenko. 2003. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine learning* 53, 1 (2003).

[26] Ryan J. Urbanowicz, Melissa Meeker, William La Cava, Randal S. Olson, and Jason H. Moore. 2018. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics* 85 (2018).

[27] Christiaan Maarten Van der Walt. 2008. *Data measures that characterise classification problems.* Ph.D. Dissertation. University of Pretoria.

[28] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013).

[29] Michel Vidal-Naquet and Shimon Ullman. 2003. Object Recognition with Informative Features and Linear Classification.. In *International Conference on Computer Vision*, Vol. 3.

[30] Philip D. Wasserman and Tom Schwartz. 1988. Neural networks. II. What are they and why is everybody so interested in them now? *IEEE expert* 3, 1 (1988).

[31] Hilde JP Weerts, Andreas C Mueller, and Joaquin Vanschoren. 2020. Importance of tuning hyperparameters of machine learning algorithms. *arXiv preprint arXiv:2007.07588* (2020).

[32] Zhou Xu, Jin Liu, Zijiang Yang, Gege An, and Xiangyang Jia. 2016. The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE).*

[33] Zheng Zhao and Huan Liu. 2007. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th international conference on Machine learning.*