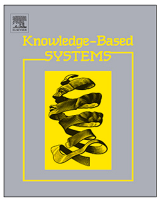
Knowledge-Based Systems 236 (2022) 107197

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



# Smart Robust Feature Selection (SoFt) for imbalanced and heterogeneous data

Gary Kee Khoon Lee [a], Henry Kasim [a], Rajendra Prasad Sirigina [b,*], Shannon Shi Qi How [a], Stephen King [c,1], Terence Gih Guang Hung [a]

[a] *Future Intelligence Technologies, Central Technology & Strategy Group, Rolls-Royce, Singapore*
[b] *Rolls-Royce@NTU Corporate Lab, Nanyang Technological University, Singapore*
[c] *School of Aerospace, Transport and Manufacturing IVHM Centre, Cranfield University, Cranfield, Bedford MK43 0AL, United Kingdom*

ARTICLE INFO

*Article history:*
Received 1 November 2020
Received in revised form 5 April 2021
Accepted 2 June 2021
Available online 30 November 2021

*Keywords:*
Class-imbalanced data
Heterogeneous features
Boosting algorithms
Feature Selection
CatBoost
H2O GBM

ABSTRACT

Designing a smart and robust predictive model that can deal with imbalanced data and a heterogeneous set of features is paramount to its widespread adoption by practitioners. By smart, we mean the model is either parameter-free or works well with default parameters, avoiding the challenge of parameter tuning. Furthermore, a robust model should consistently achieve high accuracy regardless of any dataset (imbalance, heterogeneous set of features) or domain (such as medical, financial). To this end, a computationally inexpensive and yet robust predictive model named smart robust feature selection (SoFt) is proposed. SoFt involves selecting a learning algorithm and designing a filtering-based feature selection algorithm named multi evaluation criteria and Pareto (MECP). Two state-of-the-art gradient boosting methods (GBMs), CatBoost and H2O GBM, are considered potential candidates for learning algorithms. CatBoost is selected over H2O GBM due to its robustness with both default and tuned parameters. The MECP uses multiple parameter-free feature scores to rank the features. SoFt is validated against CatBoost with a full feature set and wrapper-based CatBoost. SoFt is robust and consistent for imbalanced datasets, i.e., average value and standard deviation of log loss are low across different folds of K-fold cross-validation. Features selected by MECP are also consistent, i.e., features selected by SoFt and wrapper-based CatBoost are consistent across different folds, demonstrating the effectiveness of MECP. For balanced datasets, MECP selects too few features, and hence, the log loss of SoFt is significantly higher than CatBoost with a full feature set.

## 1. Introduction

Real-world datasets contain class-imbalanced data with a massive and heterogeneous mix of features. For example, in applications such as detecting rare events related to financial fraud, aeronautical engine failures, disease prediction, the class distribution is highly skewed [1]. Moreover, predictive models may have several parameters to tune. A typical user may lack the knowledge or resources (e.g., time and computational power) to tune the model's parameters.[2] As a result, parameter tuning may become a significant hurdle for the successful operational deployment of models. This work aims at tackling the above issues. In particular, the objective is to design a feature selection (FS) and classification model pipeline that is smart, robust, and consistent.

A smart system should have low computational complexity, and it also should have either no parameters to tune or work well with default parameters. Robustness is related to the accuracy of the model. Consistency, also known as stability, is related to bias towards the specific sampling of data used for training [2]. A consistent model's accuracy is independent of the sample of data used for training, i.e., its variation across different folds of cross-validation is minimal. Feature consistency, also known as feature stability, implies that the selected features are the same across different data sampling. Moreover, features selected by different FS algorithms are the same. Feature consistency leads to an increase in confidence in the feature set.

### 1.1. Related work

The techniques for class-imbalanced data can fit into two broad categories: data-level and algorithmic-level approaches [1,

* Corresponding author.
*E-mail addresses:* Gary.Lee@Rolls-Royce.com (G.K.K. Lee), raje0015@e.ntu.edu.sg (R.P. Sirigina).

1 This work was done while Stephen King was at Rolls-Royce, UK.

2 We refer both parameters (which are estimated from data) and hyper-parameters (which are set by heuristics) of the predictive models as parameters throughout the manuscript

https://doi.org/10.1016/j.knosys.2021.107197
0950-7051/© 2021 Elsevier B.V. All rights reserved.

3]. Sampling and data augmentation techniques fall under the category of data-level techniques. Cost-sensitive learning, where the misclassification of minority class is heavily penalized compared to that of majority class, falls under the class of algorithmic-level approach [4]. Cost-sensitive ensemble learning algorithms that use these traditional ML algorithms as base classifiers are gaining popularity. In particular, gradient boost methods (GBMs) are shown to achieve better variance and bias [5,6].

Moreover, ML algorithms that are designed for datasets with only numerical features may not work well for categorical features and heterogeneous mix of features [7]. It subsequently led to the development of packages such as XGBoost [8], CatBoost [9], H2O GBM [10], LightGBM [11]. Performance evaluation of CatBoost, H2O GBM, XGBoost, and Light GBM can be found in [7], where CatBoost is shown to achieve lower log loss than XGBoost and LightGBM. Besides, it is also shown that CatBoost quickly achieves a better area under the curve (AUC) than both XGBoost and Light GBM.

Apart from class-imbalance and heterogeneous mix of features, another challenge is the massive feature set, also known as the *curse of dimensionality*. A large feature set leads to high computational cost, overfitting, and degraded performance. Subspace projection techniques such as principal component analysis (PCA) can reduce the feature set's dimension [12]. However, the resulting principal components are not easily interpretable. The objective of this work is to retain the features in their original form. Feature selection (FS) attempts to resolve the *curse of dimensionality*, without losing interpretability of the features. The FS algorithms can be classified as the wrapper, filter, and embedded methods [13]. The learning algorithm is fixed in the wrapper and embedded methods. In the wrapper methods, the algorithms need to search over an exhaustive combination of features. Efficient search algorithms like sequential search, genetic algorithms, hill climb search algorithms are proposed [14–16]. In embedded methods, feature selection is integrated into the optimization problem of the learning algorithm. For example, decision tree based learning algorithms provide feature importance ranking as a byproduct, and these ranks can be used to select relevant features [2,17–19]. In this regard, CatBoost and H2O GBM that can also take care of class-imbalance and heterogeneous features seem to be potential solutions. However, the wrapper and embedded methods are computationally expensive, and the selected features are biased towards the learning algorithm.

On the other hand, bias is not an issue in filtering methods where the FS is independent of the learning algorithm. The filtering method involves assigning scores to features and selecting a subset of features based on those scores. For labeled data, feature scores indicate the relevance of features for a given label [20]. For unlabeled data, the feature score reflects the quality of clusters that can be formed using that feature [21,22]. Similarity-based approaches (e.g., Fisher score [23], SPEC score [22], etc.) rank the features based on their ability to preserve the distance properties. Information-theoretic approaches rely on measures such as mutual information (e.g., information gain [24], minimum redundancy maximum relevance [25]) to capture the correlation between features and labels or correlation among the features. Statistical approaches rank the features based on their ability to discriminate different data classes using statistical characteristics such as the mean. A comprehensive list of feature scores that rely on similarity metrics, information theory, and statistical methods can be found in [13, Section 2.1].

In the filtering method, multiple feature scores can select relevant features, increasing robustness and diversity. In the case of multiple feature scores, no single feature may achieve the best score across all feature selection criteria. Therefore, usually, a non-dominant subset of features, also known as the Pareto set, is selected [26]. Hancer et al. [27] proposed a multi-objective artificial bee colony optimization-based approach for filtering based feature selection. Enguerran et al. [28] proposed a Pareto frontier based FS strategy with multiple objectives. Both of these works consider information-theoretic measures such as mutual information to quantify the quality of features. Neshatian et al. [29] proposed Pareto front multivariate filtering based feature selection, where a subset of features is ranked together. Jesus et al. [30] proposed a dynamic feature selection based on information-theoretic criteria that allow different features for groups of instances.

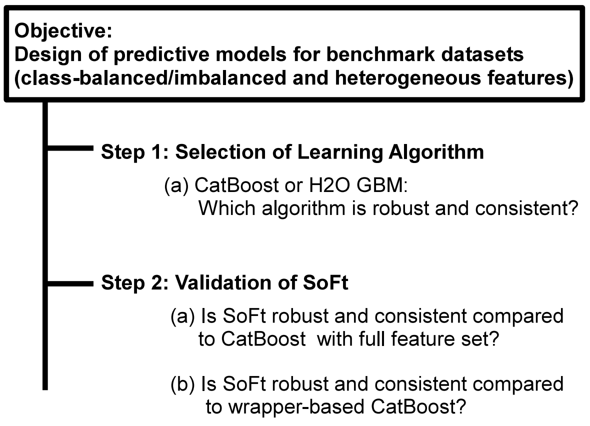

Objective:
Design of predictive models for benchmark datasets (class-balanced/imbalanced and heterogeneous features)
Step 1: Selection of Learning Algorithm
(a) CatBoost or H2O GBM: Which algorithm is robust and consistent?
Step 2: Validation of SoFt
(a) Is SoFt robust and consistent compared to CatBoost with full feature set?
(b) Is SoFt robust and consistent compared to wrapper-based CatBoost?

**Fig. 1.** Objectives (results related to Step 1 are presented in Section 3.1; results related to Step 2 are presented in Section 3.2).

### 1.2. Motivation

Interpretability and explainability of the decisions are crucial for the wider acceptability of the decisions taken by the ML algorithms by both general practitioners and regulatory bodies. There are two classes of interpretability methods: model-specific method (also known as intrinsic method) and model-agnostic method (also known post-hoc method) [31]. Model-agnostic interpretation avoids the bias arising due to the selection of a specific ML model. To this end, feature-based model-agnostic interpretable ML (IML) and explainable AI (XAI) is gaining traction [32,33]. Feature selection (FS) is critical to resolving the issues related to large dimensional datasets and for the efficient implementation of model-agnostic IML. It has motivated us to explore FS algorithms that are independent of the predictive modeling. Model-agnostic interpretability with the selected feature subset is beyond the scope of this work, and it is left for future work.

### 1.3. Objectives

The overarching objective of this work is to decouple the feature selection and learning algorithms such that the resulting model is robust yet simple for imbalanced datasets with heterogeneous features. To this end, a novel Smart robust Feature selection (SoFt) is presented. SoFt contains two components: (1) a robust state-of-the-art predictive model CatBoost; (2) a novel filtering-based FS algorithm named multi evaluation criteria and Pareto (MECP). In this section, the motivations behind the selection of the predictive model and the design of the FS algorithm are presented along with the specific objectives of this work. The objectives of this work are also summarized in Fig. 1.

#### 1.3.1. Selection of predictive model

The aim is to select a classification algorithm that achieves superior performance with imbalanced datasets and a heterogeneous set of features. CatBoost and H2O GBM are known to achieve better log loss (refer Section 2.1.4) for a heterogeneous mix of features [7]. However, hitherto, these algorithms' robustness for class-balanced and class-imbalanced datasets with a heterogeneous mix of features is unknown. Moreover, parameter tuning is a computationally-intensive and time-consuming task. If enough computing resources and time are unavailable, then a model that works well with default parameters is preferred. Hence, CatBoost and H2O GBM with both default (available at [34] and [35]) and tuned parameters are investigated.

#### 1.3.2. Design and validation of feature selection algorithm

The feature ranks provided by CatBoost can be used to select a subset of features [2]. However, typically features selected by the wrapper-based methods are biased towards the specific learning algorithm. Moreover, wrapper-based methods take a longer time than filter-based methods. Therefore, we propose filtering-based MECP algorithm, independent of the learning algorithm in CatBoost. In MECP, multiple feature scores, namely Fisher-score, reliefF-score, trace ratio, F-score, and T-score are used to rank features, and the Pareto frontier based method is proposed to select a subset of features based on these ranks. We omitted unsupervised similarity-based feature scores as they have multiple parameters to tune [22,23]. Information-theoretic feature scores are also omitted as off-the-shelf tools assume discrete datasets [36], and discretization (or quantization) techniques are not parameter-free [37]. In contrast to the state-of-the-art Pareto-based methods [27] and [28], feature scores in MECP are parameter-free. Hence, the proposed MECP algorithm is simple with low computational complexity. Our objective is to verify whether the simple algorithm SoFt (i.e., MECP+CatBoost) is robust and consistent.

### 1.4. Major contributions

First, the robustness of CatBoost and H2O GBM using $K$-fold cross-validation and stratified shuffle shift (refer Section 2.1.2) is verified. Log loss is measured for benchmark datasets with the following characteristics: imbalanced and mixture datasets, balanced and mixture datasets, and balanced and numerical datasets (refer Table 1 for more details). It is shown that CatBoost achieves lower log loss than H2O GBM for most datasets with the default and tuned parameters (refer Table 6). Moreover, we observed that the features ranked high by CatBoost and H2O GBM are similar across different splits of the data (refer Fig. 3), reflecting the feature stability of these algorithms. It has motivated us to select CatBoost as the predictive model for the SoFt algorithm.

Next, we evaluate the log loss of SoFt and compare it against CatBoost with full feature set (i.e., without FS) and wrapper-based CatBoost. Note that in the case of wrapper-based CatBoost, the selected feature subset's size is constrained to be the same as that of SoFt. SoFt is shown to be consistent, i.e., the standard deviation of log loss across all random splits is low (refer Table 7). For all datasets, it is shown that wrapper-based CatBoost and SoFt achieves similar log loss. For imbalanced datasets, the percentage difference in the log loss between SoFt and CatBoost with full feature set is low, i.e., proposed SoFt is robust for imbalanced datasets. Moreover, features selected by SoFt and wrapper-based CatBoost are consistent, i.e., the features selected by them are similar across different random splits (refer Fig. 4 and Table 9). SoFt is also shown to be faster than both CatBoost with the full feature set and wrapper-based CatBoost (refer Table 8). On the other hand, for balanced datasets, proposed MECP selects too few features, leading to an increase in log loss for SoFt and wrapper-based CatBoost (refer Table 7).

In a nutshell, the contributions of this work are two-fold: (1) A robust practitioner-centric (i.e., parameter-free) feature selection algorithm MECP to select interpretable features. (2) Validation through empirical evaluations that the decoupled MECP-based feature selection achieves comparable performance as that of wrapper-based feature selection for imbalanced and heterogeneous datasets that are commonly seen in real world applications.

## 2. Experimental setup and methods

We conducted three experiments. (A) Analysis of predictive modeling algorithm with full feature sets; it helps us select a learning algorithm for SoFt. (B) Analysis of the wrapper-based predictive model; it is the benchmark to validate proposed SoFt. (C) Design and validation of filtering-based SoFt. Next, a detailed description of workflows is presented.

### 2.1. Predictive modeling with full feature set

The workflow of predictive model without FS is shown in Fig. 2(a), and a detailed description of the blocks in Fig. 2(a) is presented below.

#### 2.1.1. Dataset details/characteristics and preprocessing

A quick summary of the datasets used for validation of the algorithms is shown in Table 1. A variety of datasets with characteristics such as class-balanced and class-imbalanced, homogeneous (only numerical features), and heterogeneous mix (numerical and categorical) of features are considered. These datasets are collected from the benchmark data [40]. Additional balanced data of each attribute type are collected from the UCI ML Repository [41] and KDD Cup 2009 [42]. We investigate two aspects of the data characteristics: (1) the class balance/imbalance of the dataset and (2) the number of categorical/numerical features in the dataset. Note that if the percentage of a class is larger than 75%, we consider the dataset as class-imbalanced. The Adult, Australian Credit, and Heart datasets have majority categorical features, while Bank and German Credit datasets have close to 50% of categorical features. The motivation for using several datasets of different characteristics is to avoid the bias of the model performance. Moreover, we want to evaluate if certain predictive and FS methods are more favorable to specific data characteristics.

The preprocessing tasks involve filling up of missing values based on imputation for both numerical and categorical features. For categorical features, missing values are replaced with a special value, i.e., the missing values as a special category. While for numerical features, missing values are replaced with zeros, and a binary dummy feature for each imputed feature is added [38]. We acknowledge that there are many strategies for dealing with missing data. However, the selection of an optimal replacement strategy is out of scope for this work, and hence a simple scheme is implemented to maintain focus on feature selection.

#### 2.1.2. Random data split for training and testing

We perform stratified shuffle shift (SSS), where each dataset is randomly split into a training set (80%) and test set (20%), to perform $K$-fold cross-validation. Hence, $K$ random splits that preserve the percentage of samples related to each class are passed to the predictive model. We obtain the point estimate of the learning algorithm's performance measure by averaging its value across splits. We also evaluate the sensitivity of the learning algorithms to the random splits. To this end, the confidence level in the point estimate is measured through standard deviation. A large standard deviation indicates that the algorithm is not

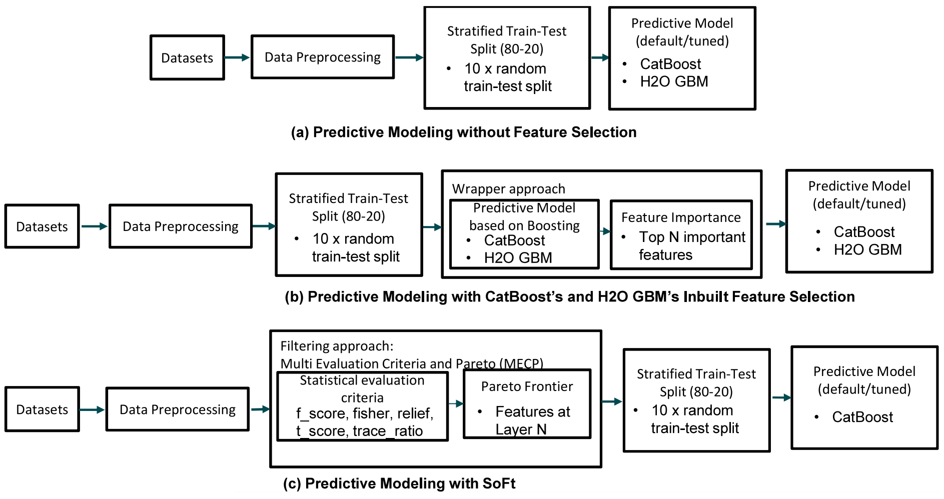

Datasets
Data Preprocessing
Stratified Train-Test Split (80-20)
• 10 x random train-test split
Predictive Model (default/tuned)
• CatBoost
• H2O GBM
(a) Predictive Modeling without Feature Selection
Wrapper approach
Predictive Model based on Boosting
• CatBoost
• H2O GBM
Feature Importance
• Top N important features
(b) Predictive Modeling with CatBoost's and H2O GBM's Inbuilt Feature Selection
Filtering approach: Multi Evaluation Criteria and Pareto (MECP)
Statistical evaluation criteria
f_score, fisher, relief, t_score, trace_ratio
Pareto Frontier
• Features at Layer N
(c) Predictive Modeling with SoFt

**Fig. 2.** Workflows of Predictive Modeling: refer Table 1 for dataset details; preprocessing involves dealing with missing values (refer Section 2.1.1 and [38]); stratified shuffle split is performed for $K$-fold cross-validation (refer Section 2.1.2); details related to predictive models CatBoost and H2O GBM are in Section 2.1.3; parameter tuning is done with H2O.ai [39]; wrapper and filtering-based FS are presented in Section 2.2; CatBoost achieves lower log loss than H2O GBM (refer Table 6) and hence CatBoost is selected over H2O GBM for SoFt.

**Table 1**
Details of the datasets (I, B, N, and M denote class-imbalanced, class-balanced, only numerical features, and both numerical and categorical features, respectively).[2]

| Data | # of Features | # of Categorical | # of Numerical | # of Samples | Ratio of target class |
|---|---|---|---|---|---|
| Adult (I,M) | 14 | 8 | 6 | 48842 | 76:24 |
| Appet (I,M) | 230 | 40 | 190 | 50000 | 98:2 |
| Churn (I,M) | 230 | 40 | 190 | 50000 | 93:7 |
| Upsel (I,M) | 230 | 40 | 190 | 50000 | 93:7 |
| Australian Credit (B,M) | 14 | 8 | 6 | 690 | 56:44 |
| Bank (B,M) | 21 | 10 | 11 | 9280 | 50:50 |
| German Credit (B,M) | 20 | 7 | 13 | 1000 | 70:30 |
| Heart (B,M) | 13 | 8 | 5 | 303 | 54:46 |
| Breast Cancer Wisconsin (B, N) | 30 | 0 | 30 | 569 | 63:37 |
| Higgs (B,N) | 28 | 0 | 28 | 15000 | 50:50 |
| QSAR (B,N) | 41 | 0 | 41 | 1055 | 66:34 |
| Vertebral (B,N) | 6 | 0 | 6 | 310 | 68:32 |

robust, i.e., highly sensitive to the samples of the dataset used for training, leading to a decrease in the confidence level on the average value of predictive model performance.

### 2.1.3. *Predictive modeling*

Both CatBoost and H20 GBM are gradient boosting algorithms. A detailed description of these protocols can be found in [9] and [44], respectively, and they are omitted here for the sake of brevity. Only the differences in architectures of these two algorithms are presented here. CatBoost uses oblivious decision trees as base predictors, and it implements a novel ordered boosting algorithm to avoid target leakage [7]. On the other hand, H2O GBM utilizes a forward learning ensemble approach, and it utilizes all features to build decision trees sequentially in a distributed manner [10]. We evaluate the performance of the predictive models with both default parameter values and tuned parameter values. Exhaustive search is done for the number of trees with the maximum number of trees set as 150. This process is carried out 50 times, and the lowest log loss is recorded. The parameter space for tuning and default parameters is the same as that in [39], and there is randomness associated with the tuning of parameter space of CatBoost. Specific details related to parameters are presented in Section 3.

### 2.1.4. *Performance measure*

Gradient boosting method calculates the nonlinear predictor $\hat{f}(\mathbf{x}) \in \mathcal{F}$ as $\hat{f} = \arg\min_{f\in\mathcal{F}} \mathcal{L}(f(X), Y)$, where $\mathbf{x}$ is the feature vector, $\mathcal{F}$ is the space of weak learners, $\mathcal{L}(.)$ is the loss function, $X$ is the feature matrix, and $Y$ is the target vector [45]. Accuracy is not an appropriate metric for imbalanced datasets. Rather, the AUC is one of the widely used metrics to evaluate the performance of models for imbalanced datasets. It provides a single measure for trade-off between true positive rate and false positive rate via area under receiver operating characteristics curve. However, AUC is a non-differentiable function. It cannot be used in stochastic gradient boosting methods. Instead, a differentiable surrogate loss function log loss, which is consistent with AUC, is used in gradient boosting methods [46]. The log loss function is defined as $\mathcal{L}(f(X), Y) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + \exp(-f(\mathbf{x_i})y_i))$, where $x_i$ and $y_i$ are the feature vector and target of $i$th instance, respectively. If target and predicted class distributions of a model matches, then the probability that the model makes correct classification is high, which leads to low log loss, and model is good. Otherwise, log loss is high and model is bad.

[2] We assume that the raw data is converted into a set of features. Though this work considers only non-time series datasets, the analysis can be extended to feature-based time series classification. For the time series datasets that are collected using sensors mounted on industrial machines, we need an additional block called feature engineering (after preprocessing in Fig. 2) to convert raw time series data into features [43].

### 2.2. Predictive modeling with feature selection

Two FS methods, wrapper method (both CatBoost and H2O GBM) and filter (MECP) approaches, are investigated. Data pre-processing, stratified shuffle shift, parameter tuning, and performance measure are the same as those described in Section 2.1. Hence, the description of these blocks is omitted here.

#### 2.2.1. Predictive modeling with wrapper-based FS

Both CatBoost and H2O GBM provide the ranks to features based on their importance. For these gradient boosting models, the feature importance value is based on whether the split for each node depends on the particular feature and the weighted variance over all trees [9]. As shown in Fig. 2, $K$-fold SSS is performed on datasets. The feature rank is evaluated for each fold using CatBoost/H2O GBM. The final rank of each feature is the average of its rank across all folds. For example, if a feature is assigned rank one in all folds, its average rank across all folds is one. We use wrapper-based CatBoost to validate the proposed SoFt algorithm. Therefore, the number of features selected by wrapper-based CatBoost is constrained to be the same as that of SoFt to ensure a fair comparison between SoFt and wrapper-based CatBoost.

#### 2.2.2. Predictive modeling with SoFt (MECP-based FS)

A $K$-fold cross-validation based on SSS is used to avoid bias or over-fitting the selected features towards a specific dataset sample. The number of train-test (80%–20%) splits are determined based on the number of minority class samples, with the minimum and maximum number of splits are two and ten, respectively. In this work, the categorical features are encoded into numerical values using Python's Scikit-learn preprocessing block. Such a transformation enables the multi-evaluation criteria for each fold of the training dataset based on five metrics: t-score, F-score, Fisher-score, reliefF-score, and trace ratio. None of these feature scores have parameters to tune. These metrics are used to rank the numerical features based on their ability to preserve the similarity of instances within each class and discriminate against the instances that belong to different classes.

Let $N$, and $n_c$ be the total number of features, and total number of features in $c$th class, respectively. Let $f_i$, $f_{i,c}$, $f_{i,c}^k$, $\mu_i$ and $\sigma_i$, and $\mu_{i,c}$ and $\sigma_{i,c}$, where $i = \{1, \dots, N\}$, $c = \{1, 2\}$, be the $i$th feature, $i$th feature in $c$th class, $i$th feature that belongs to $k$th instance of $c$th class, mean and variance of $i$th feature, and mean and variance of $i$th feature in $c$th class, respectively, of a binary classification problem. Let $F(f_i)$, $T(f_i)$, $\mathit{Fisher}(f_i)$, $\mathit{reliefF}(f_i)$, and $\mathit{trace\_ratio}(f_i)$ be the F-score, T-score, Fisher-score, reliefF-score, and trace ratio of the feature $f_i$, respectively. $F(f_i)$ is defined as [47]

$$F\left(f_i\right) = \frac{\sum_{c=1}^{2}\left(\mu_{i,c} - \mu_i\right)^2}{\sum_{c=1}^{2}\frac{1}{n_c - 1}\sum_{k=1}^{n_c}\left(f_{i,c}^k - \mu_{i,c}\right)^2} \tag{1}$$

The F-score is used to determine how well a feature can discriminate two sets of real numbers, and large F-score means the discrimination power of the feature is high. The t-score of a feature is defined as [13]

$$T\left(f_i\right) = \frac{\left|\mu_{i,1} - \mu_{i,2}\right|}{\sqrt{\frac{\sigma_{i,1}^2}{n_1} + \frac{\sigma_{i,2}^2}{n_2}}} \tag{2}$$

A feature with large T-score means average values of two classes in a binary classification problem are statistically different, i.e., the feature is important. The Fisher-score of a feature $f_i$ is defined as [13]

$$\mathit{Fisher}(f_i) = \frac{\sum_{c=1}^{2} n_c\left(\mu_{i,c} - \mu_i\right)^2}{\sum_{c=1}^{2} n_c\sigma_{i,c}^2} \tag{3}$$

**Algorithm 1** Multi Evaluation Criteria and Pareto (MECP)

```
Input:
    $\mathbb{D} \in \mathbb{R}^{I\times N}$: Dataset ($I$ and $N$ denote number of instances and features, respectively)
Output:
    $\mathbb{F}_S$: Ordered unique feature subset at Pareto layer 0
 1: procedure MECP($\mathbb{D}$: Data Set)
 2:     $\mathbb{F}_0 \leftarrow \varnothing$                      ▷ Set of features in Pareto layer 0
 3:     for $k = 1$ to $K$ do                    ▷ $K$-fold cross-validation
 4:         $\mathbb{D}_{K-1} = SSS(D)$          ▷ Data from $K-1$ folds of Stratified Shuffle Shift (SSS)
 5:         for $s = 1$ to $S$ do                ▷ Five feature scores ($S = 5$)
 6:             for $i = 1$ to $N$ do
 7:                 $f_{i,k} \leftarrow \mathbb{D}_{K-1}(i)$      ▷ $i$-th feature vector
 8:                 $G[i] \leftarrow \phi_s(f_{i,k})$        ▷ Calculation of feature score of $i$-th feature at $k$-th fold
 9:             end for
10:             $\hat{G} \leftarrow SORT(G)$           ▷ Sort features based on $s$-th feature score
11:             $\mathbb{F}_0 \leftarrow \mathbb{F}_0 \cup \hat{G}[1]$         ▷ Select top feature
12:         end for
13:     end for
14:     $\mathbb{F}_S \leftarrow$ RANK($\mathbb{F}_0$)      ▷ Rank the features based on number of times it appeared in $\mathbb{F}_0$
15: end procedure
```

A feature with high Fisher-score indicates that the feature is important. The algorithms to calculate the $\mathit{ReliefF}(f_i)$ and $\mathit{trace\_ratio}(f_i)$ can be found in [20, Section 2] and [13, Section 2.1.4], respectively, and we omitted these details for brevity.

The pseudo code for the proposed MECP is presented in Algorithm 1. Let $\phi_s(.)$, $s = \{1, \dots, 5\}$, denote the functions $F(.)$, $t(.)$, $\mathit{Fisher}(.)$, $\mathit{reliefF}(.)$, and $\mathit{trace\_ratio}(.)$. Let $K = 10$ and $S = 5$ denote the number of folds for cross-validation and number of feature scores, respectively. At each fold, the feature scores are calculated and arranged in decreasing order of their rank. Top feature from each feature score is added to feature set $\mathbb{F}_0$ of Pareto layer 0. The features in $\mathbb{F}_0$ are ranked based on the frequency with which the features appeared. For example, top ranked feature at layer 0 is selected most number of times by feature scores at each fold. The algorithm returns the feature subset $\mathbb{F}_S$, that contains features and their ranks. Finally, in each layer, the filtering technique is used to eliminate features that are highly correlated (based on Pearson correlation). In Algorithm 1, we omitted the description of SORT and RANK functions for brevity.

## 3. Experiments and results

As shown in Fig. 1, we present the analysis of CatBoost (version 0.22) and H2O GBM (version 3.28.0.4), and later we show the results related to the validation of SoFt. The analysis is conducted for three categories of data: imbalanced (I) and mixture (M) features, balanced (B) and mixture (M) features, and balanced (B) and numerical (N) features.

### 3.1. Analysis of CatBoost and H2O GBM

#### 3.1.1. Investigation of robustness

The robustness is evaluated for both default (refer Tables 2 and 3) tuned (refer Tables 4 and 5) parameters [38]. The hyper-parameter tuning for both CatBoost and H2O GBM is based on

**Table 2**
Default parameters of CatBoost.

| Parameter | Value |
|---|---|
| learning_rate | 0.03 |
| depth | 6 |
| fold_len_multiplier | 2 |
| rsm | 1.0 |
| border_count | 128 |
| l2_leaf_reg | 3 |
| leaf_estimation_method | Newton |
| ctr_description | [‘Borders’,‘Counter’] |
| used_ram_limit | 100000000000 |
| loss_function | Logloss |
| iterations | 150 |

**Table 3**
Default parameters of H20 GBM.

| Parameter | Value |
|---|---|
| learn_rate | 0.1 |
| max_depth | 5 |
| sample_rate | 1 |
| col_sample_rate | 1 |
| col_sample_rate_change_per_level | 1 |
| min_split_improvement | exp(−5) |
| min_rows | 10 |
| histogram_type | ‘auto’ |

**Table 4**
Tuning parameters for catboost.

| Parameter | Value |
|---|---|
| learn_rate | Log-Uniform $[e^{-7}, 1]$ |
| max_depth | qUniform [2, 10, 1] |
| sample_rate | Uniform [0.5, 1] |
| col_sample_rate | Uniform [0.5, 1] |
| col_sample_rate_change_per_level | Uniform [0.5, 1] |
| min_split_improvement | Log-Uniform [−16, 0] |
| min_rows | Log-Uniform [0, 5] |
| histogram_type | {‘uniform_adaptive’,‘random’, ‘quantiles_global’,‘round_robin’} |

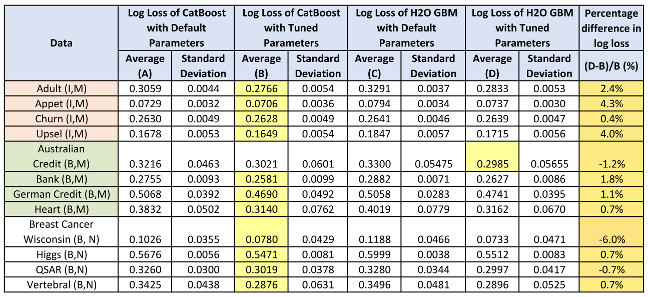
**Table 5**
Tuning parameters for H20 GBM.

| Parameter | Value |
|---|---|
| learn_rate | Log-Uniform $[e^{-7}, 1]$ |
| max_depth | qUniform[2, 10, 1] |
| sample_rate | Uniform [0.5, 1] |
| col_sample_rate | Uniform [0.5, 1] |
| col_sample_rate_change_per_level | Uniform[0.5, 1] |
| min_split_improvement | Log-Uniform [−16, 0] |
| min_rows | Log-Uniform [0, 5] |
| histogram_type | {‘uniform_adaptive’,‘random’, ‘quantiles_global’,‘round_robin’}) |

grid search functionality provided in the respective libraries. The grid search function searches over specific parameter values for a model based on the best loss function i.e, log-loss. The parameter ranges for grid search are provided in Tables 4 and 5, respectively, for the CatBoost and H2O GBM. The objective is to determine the robustness (i.e., log loss) and consistency (i.e., the variation of log loss across different folds of cross-validation) of CatBoost and H2O GBM. To this end, the average value and standard deviations of log loss are evaluated with ten different random splits. The corresponding results are presented in Table 6 for both default and tuned parameters. CatBoost and H2O GBM achieve similar log loss for default parameter settings, with CatBoost achieving slightly better log loss than H2O GBM. It can also be seen that both models are not sensitive to random train-test split, as the standard deviations are low.

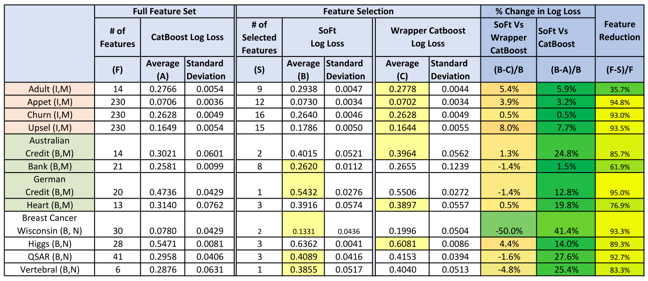
The performance of tuned CatBoost and tuned H2O GBM is comparable for all datasets. For balanced datasets, CatBoost and H2O GBM achieve similar log loss, except for Breast Cancer Wisconsin dataset that has around 6% difference. CatBoost performs slightly better for imbalanced datasets than H2O GBM with a maximum difference of around 4% difference. The algorithms are also consistent with tuned parameters as the standard deviation of log loss is low. In summary, both CatBoost and H2O GBM are robust, and CatBoost is preferred over H2O GBM even when the users do not have resources to tune the model.

**Table 6**
Comparison of log loss of CatBoost and H2O GBM with full feature set for both default parameters and tuned parameters; Dataset details can be found in Table 1; average log loss of CatBoost is lower than H2O GBM for majority of datasets and variation of log loss (i.e., standard deviation) across different folds of $K$-fold cross-validation is low.



| Data | Log Loss of CatBoost with Default Parameters | | Log Loss of CatBoost with Tuned Parameters | | Log Loss of H2O GBM with Default Parameters | | Log Loss of H2O GBM with Tuned Parameters | | Percentage difference in log loss |
|---|---|---|---|---|---|---|---|---|---|
| | Average (A) | Standard Deviation | Average (B) | Standard Deviation | Average (C) | Standard Deviation | Average (D) | Standard Deviation | (D-B)/B (%) |
| Adult (I,M) | 0.3059 | 0.0044 | 0.2766 | 0.0054 | 0.3291 | 0.0037 | 0.2833 | 0.0053 | 2.4% |
| Appet (I,M) | 0.0729 | 0.0032 | 0.0706 | 0.0036 | 0.0794 | 0.0034 | 0.0737 | 0.0030 | 4.3% |
| Churn (I,M) | 0.2630 | 0.0049 | 0.2628 | 0.0049 | 0.2641 | 0.0046 | 0.2639 | 0.0047 | 0.4% |
| Upsel (I,M) | 0.1678 | 0.0053 | 0.1649 | 0.0054 | 0.1847 | 0.0057 | 0.1715 | 0.0056 | 4.0% |
| Australian Credit (B,M) | 0.3216 | 0.0463 | 0.3021 | 0.0601 | 0.3300 | 0.05475 | 0.2985 | 0.05655 | -1.2% |
| Bank (B,M) | 0.2755 | 0.0093 | 0.2581 | 0.0099 | 0.2882 | 0.0071 | 0.2627 | 0.0086 | 1.8% |
| German Credit (B,M) | 0.5068 | 0.0392 | 0.4690 | 0.0492 | 0.5058 | 0.0283 | 0.4741 | 0.0395 | 1.1% |
| Heart (B,M) | 0.3832 | 0.0502 | 0.3140 | 0.0762 | 0.4019 | 0.0779 | 0.3162 | 0.0670 | 0.7% |
| Breast Cancer Wisconsin (B, N) | 0.1026 | 0.0355 | 0.0780 | 0.0429 | 0.1188 | 0.0466 | 0.0733 | 0.0471 | -6.0% |
| Higgs (B,N) | 0.5676 | 0.0056 | 0.5471 | 0.0081 | 0.5999 | 0.0038 | 0.5512 | 0.0083 | 0.7% |
| QSAR (B,N) | 0.3260 | 0.0300 | 0.3019 | 0.0378 | 0.3280 | 0.0344 | 0.2997 | 0.0417 | -0.7% |
| Vertebral (B,N) | 0.3425 | 0.0438 | 0.2876 | 0.0631 | 0.3496 | 0.0481 | 0.2896 | 0.0525 | 0.7% |

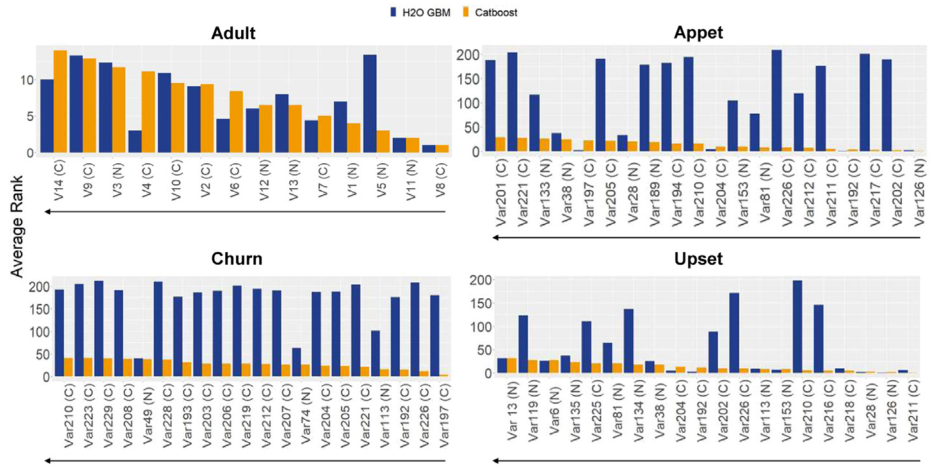
**Table 7**
Comparison of log loss of CatBoost (with full feature set), SoFt, and wrapper-based CatBoost: all algorithms have low log loss (i.e., robust) and low standard deviation (i.e., consistent across random splits); for imbalanced (I) datasets, percentage increase in log loss of SoFt compared to CatBoost with full feature set is minimal; for balanced (B) datasets, percentage increase in log loss is high with feature selection as SoFt selects too few features.



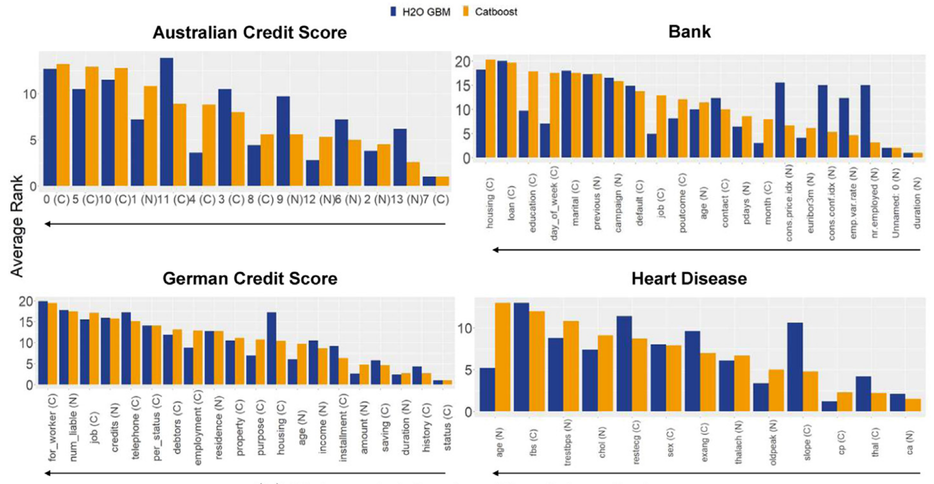
| | Full Feature Set | | | Feature Selection | | | | | % Change in Log Loss | | Feature Reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | # of Features | CatBoost Log Loss | | # of Selected Features | SoFt Log Loss | | Wrapper Catboost Log Loss | | SoFt Vs Wrapper CatBoost | SoFt Vs CatBoost | |
| | (F) | Average (A) | Standard Deviation | (S) | Average (B) | Standard Deviation | Average (C) | Standard Deviation | (B-C)/B | (B-A)/B | (F-S)/F |
| Adult (I,M) | 14 | 0.2766 | 0.0054 | 9 | 0.2938 | 0.0047 | 0.2778 | 0.0044 | 5.4% | 5.9% | 35.7% |
| Appet (I,M) | 230 | 0.0706 | 0.0036 | 12 | 0.0730 | 0.0034 | 0.0702 | 0.0034 | 3.9% | 3.2% | 94.8% |
| Churn (I,M) | 230 | 0.2628 | 0.0049 | 16 | 0.2640 | 0.0046 | 0.2628 | 0.0049 | 0.5% | 0.5% | 93.0% |
| Upsel (I,M) | 230 | 0.1649 | 0.0054 | 15 | 0.1786 | 0.0050 | 0.1644 | 0.0055 | 8.0% | 7.7% | 93.5% |
| Australian Credit (B,M) | 14 | 0.3021 | 0.0601 | 2 | 0.4015 | 0.0521 | 0.3964 | 0.0562 | 1.3% | 24.8% | 85.7% |
| Bank (B,M) | 21 | 0.2581 | 0.0099 | 8 | 0.2620 | 0.0112 | 0.2655 | 0.1239 | -1.4% | 1.5% | 61.9% |
| German Credit (B,M) | 20 | 0.4736 | 0.0429 | 1 | 0.5432 | 0.0276 | 0.5506 | 0.0272 | -1.4% | 12.8% | 95.0% |
| Heart (B,M) | 13 | 0.3140 | 0.0762 | 3 | 0.3916 | 0.0574 | 0.3897 | 0.0557 | 0.5% | 19.8% | 76.9% |
| Breast Cancer Wisconsin (B, N) | 30 | 0.0780 | 0.0429 | 2 | 0.1331 | 0.0436 | 0.1996 | 0.0504 | -50.0% | 41.4% | 93.3% |
| Higgs (B,N) | 28 | 0.5471 | 0.0081 | 3 | 0.6362 | 0.0041 | 0.6081 | 0.0086 | 4.4% | 14.0% | 89.3% |
| QSAR (B,N) | 41 | 0.2958 | 0.0406 | 3 | 0.4089 | 0.0416 | 0.4153 | 0.0394 | -1.6% | 27.6% | 92.7% |
| Vertebral (B,N) | 6 | 0.2876 | 0.0631 | 1 | 0.3855 | 0.0517 | 0.4040 | 0.0513 | -4.8% | 25.4% | 83.3% |

### 3.1.2. Investigation of feature consistency

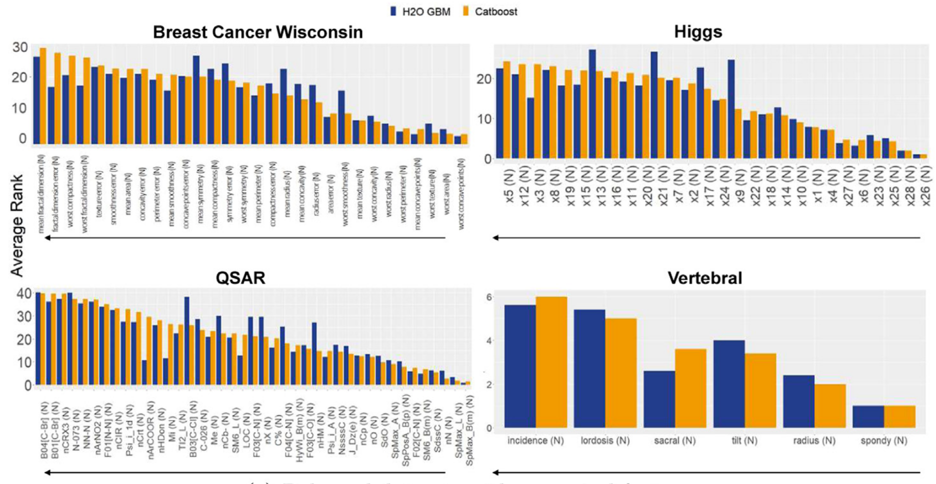
The motivation is to identify the similarity among the features that are selected by CatBoost and H2O GBM. If the feature rank is low in both models, it indicates that the feature is important. The average rank is calculated by taking the average of the rank across the ten random splits. The average rankings of the features are shown in Fig. 3. The $X$-axis represents the list of the features and the feature type (Categorical and Numerical). The number of features on the $X$-axis is limited to 20 for ease of visualization. $Y$-axis represents the average rank of the feature, where the average derived from the sum of all the feature ranking in ten random splits divided by ten. The range of average rank of a feature lies between one (10/10 assuming in all ten splits the feature rank is always one) and the total number of features (for example, the maximum rank is 230 for Appet, Churn, and Upsel datasets). Hence, the low average rank (right side of the graph) indicates an important feature. The orange and blue bars represent the ranks provided by CatBoost and H2O GBM, respectively. The features on $X$-axis are ordered based on the rank provided by CatBoost, with features on the right that have a low rank, i.e., the features on the right are selected as important by CatBoost.



(a) Imbalanced datasets with mixture features



(b) Balanced datasets with mixture features



(c) Balanced datasets with numerical features

**Fig. 3.** Consistency of feature ranks across folds of $K$-fold cross-validation and learning algorithms: the features are arranged from left to right based on the ranks provided by CatBoost (left most feature has lowest/top rank); (C) and (N) indicate categorical and numerical features, respectively; categorical feature V8 in Adult dataset is ranked top by both algorithms across all folds, therefore feature V8 is important; except for Appet and Churn datasets, features in majority of datasets are consistent across all folds, and they are also consistent across both algorithms (see Section 3.1.2 for details.).

The majority of the selected features by both models are consistent across all datasets, except for Appet and Churn datasets. For example, in Fig. 3(a), the categorical feature V8 in the Adult dataset is selected as important by both CatBoost and H2O GBM consistently. We mean V8 received the lowest rank across all the train test splits, making it the most important feature. On the other hand, the numerical feature V5 in the Adult dataset is ranked low by CatBoost while it received a large rank by H2O

**Table 8**
Comparison of computational times for predictive models: improvement in computational time with SoFt is insignificant with small number of features; SoFt saves significant time for datasets with large number of features.

| Data | Tuned Catboost | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | All Features | | | SoFt | | | Difference | |
| | Log Loss | # of Features | Time (Seconds) | Log Loss | # of Features | Time (Seconds) | Log Loss | Time |
| Adult (I,M) | 0.2766 | 14 | 343.27 | 0.2938 | 9 | 273.43 | 5.9% | -25.5% |
| Appet (I,M) | 0.0706 | 230 | 1198.71 | 0.0730 | 12 | 596.24 | 3.2% | -101.0% |
| Churn (I,M) | 0.2628 | 230 | 518.24 | 0.2640 | 16 | 187.56 | 0.5% | -176.3% |
| Upsel (I,M) | 0.1649 | 230 | 1249.90 | 0.1786 | 15 | 462.76 | 7.7% | -170.1% |
| Australian Credit (B,M) | 0.3021 | 14 | 141.11 | 0.4015 | 2 | 124.60 | 24.8% | -13.3% |
| Bank (B,M) | 0.2581 | 21 | 105.20 | 0.2620 | 8 | 73.74 | 1.5% | -42.7% |
| German Credit (B,M) | 0.4736 | 20 | 192.75 | 0.5432 | 1 | 131.06 | 12.8% | -47.1% |
| Heart (B,M) | 0.3140 | 13 | 183.89 | 0.3916 | 3 | 130.11 | 19.8% | -41.3% |
| Breast Cancer Wisconsin (B, N) | 0.0780 | 30 | 120.41 | 0.1331 | 2 | 99.70 | 41.4% | -20.8% |
| Higgs (B,N) | 0.5471 | 28 | 211.66 | 0.6362 | 3 | 131.43 | 14.0% | -61.0% |
| QSAR (B,N) | 0.2958 | 41 | 158.15 | 0.4089 | 3 | 135.36 | 27.6% | -16.8% |
| Vertebral (B,N) | 0.2876 | 6 | 134.00 | 0.3855 | 1 | 123.28 | 25.4% | -8.7% |

**Table 9**
Number of categorical features selected by SoFt (i.e., MECP) and CatBoost: MECP selects more categorical features than CatBoost.

| Data | All Features | | | Feature Selection | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | # Features | MECP | | CatBoost | |
| | # Features | Categorical | % Categorical | | Categorical | % Categorical | Categorical | % Categorical |
| Adult (I,M) | 14 | 8 | 57% | 9 | 5 | 56% | 4 | 44% |
| Appet (I,M) | 230 | 40 | 17% | 12 | 10 | 83% | 9 | 75% |
| Churn (I,M) | 230 | 40 | 17% | 16 | 13 | 81% | 13 | 81% |
| Upsel (I,M) | 230 | 40 | 17% | 15 | 12 | 80% | 8 | 53% |
| Australian Credit (B,M) | 14 | 8 | 57% | 2 | 2 | 100% | 1 | 50% |
| Bank (B,M) | 21 | 10 | 48% | 8 | 5 | 63% | 1 | 13% |
| German Credit (B,M) | 20 | 7 | 35% | 1 | 1 | 100% | 1 | 100% |
| Heart (B,M) | 13 | 8 | 62% | 3 | 2 | 67% | 2 | 67% |

GBM. In H2O GBM, the feature did not receive low rank across all data splits. The features that are consistently ranked low by both packages can be used for tasks such as interpretation/explanation of ML algorithms' decisions with an increased confidence level.

### 3.2. Validation of SoFt

The robustness of the filter-based approach, SoFt, is compared with that of the wrapper approach. Only the wrapper-based tuned CatBoost is considered as it outperforms H2O GBM in most datasets. The number of selected features for the wrapper-based CatBoost experiment is constrained to be the same as that of SoFt to ensure a fair comparison. We investigated log loss and similarity of the selected features by CatBoost and SoFt. Table 7 shows the log loss of CatBoost with the full feature set, wrapper-based CatBoost, and SoFt. The percentage reduction in the number of features is also shown in Table 7. The computational times of these approaches are presented in Table 8. A detailed explanation of these results is presented subsequently for different datasets.

#### 3.2.1. Imbalanced and mixture (I, M)

For (I, M) datasets, from Table 7, it can be seen that the percentage of feature reduction is in the range of 35% (Adult dataset) to 94% (Appet dataset). CatBoost with a full feature set as input outperforms SoFt by a maximum of only 8% in log loss. Therefore, the proposed SoFt is robust, even with a huge reduction in the feature set and computational load. Besides, feature reduction decreases the computation time, as shown in Table 8. To be specific, the computational time is reduced to half compared to that of CatBoost with a full feature set. Note that in the case of the Adult dataset, the computational time reduction is low for SoFt as the number of features in the dataset are low. We also validated the sensitivity of the random train-test split. It can be seen that the standard deviations of the methods are small, i.e., wrapper-based CatBoost and SoFt are consistent across different folds.

Finally, we investigate the feature consistency, i.e., whether the features selected by CatBoost and SoFt are the same or not. To this end, the features selected by SoFt and also ranked top by wrapper-based CatBoost are shown in Fig. 4(a). The features selected by MECP at layer 0 are plotted on $X$-axis. In wrapper-based CatBoost, the selected feature subset is unique in each fold of $K$-fold cross-validation. The bar's height corresponding to each feature in Fig. 4 represents the number of folds the CatBoost selects the feature. The color from green to red represents the average rank of the features from the CatBoost wrapper method. The average rank value is in percentile, where zero represents a top rank. It is done to standardize the color-coding across datasets that have a different number of features. For example, Adult dataset has 14 features; if a feature's ranks are 1, 1, 1, 3, 3, 3, 4, 5, 5, 5 in 10-fold cross-validation, then the feature's average rank is 3.1, and its percentile is 22.1. Fig. 4(a) shows that the top features (left bar in X-axis) selected by SoFt are the same as those of the CatBoost wrapper approach. For example, in Appet dataset, numerical feature Var126 is selected by MECP and also ranked high by wrapper-based CatBoost, while categorical feature Var204 is selected by MECP but not selected in any of the folds in wrapper-based CatBoost; Further down the $X$-axis, the wrapper-based CatBoost ranking for the features selected by MECP gradually decreases. MECP selects more categorical features than CatBoost wrapper, and the ranking for these features is set to zero as they are not in the top N feature list of wrapper-based CatBoost approach. Table 9, shows that MECP selects more categorical features than wrapper-based CatBoost approach, especially for the Upsel dataset. MECP uses multi-criteria evaluation with equal weightage, and one of the statistical methods (t-score) in MECP is scoring categorical features better than numerical features.

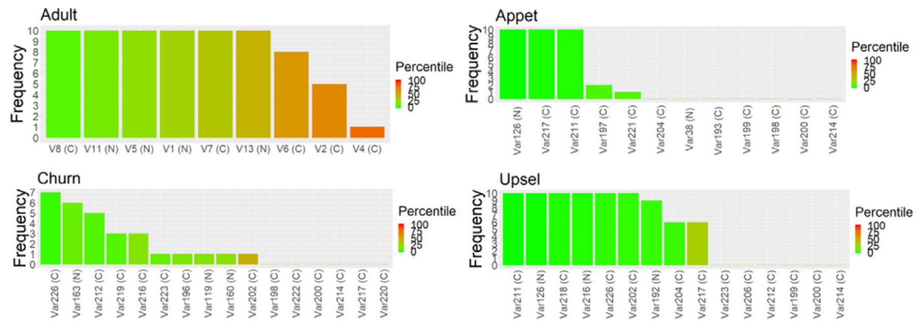
#### 3.2.2. Balanced and mixture (B, M)

For (B, M) datasets, Table 7 shows that the percentage reduction in the size of the feature set by SoFt lies in the range of 62% to 95%. It can also be seen that CatBoost with a full feature set achieves better log loss than SoFt. The maximum percentage increase in log loss for SoFt compared to CatBoost with full feature set is around 25%. It is because the SoFt selects too few features (one to three features) that are insufficient to perform the respective classification task. It is also the case for the wrapper-based CatBoost as CatBoost is constrained to select the same number of features as that of SoFt. A potential solution to this problem is to select features from layer $N$, where $N \geq 1$. The log loss values of SoFt and wrapper-based CatBoost are comparable for (B,M) datasets with the percentage difference in the range of $-1.4\%$ to 1.3%. Moreover, all algorithms are consistent from log loss perspective as the standard deviations of log loss are very low.

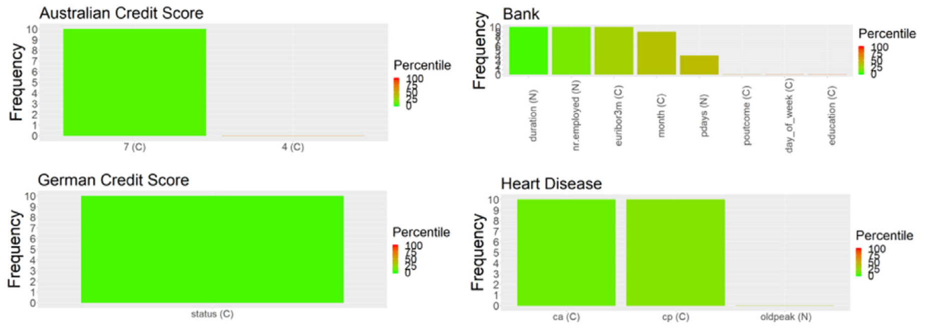
Next, the feature consistency of CatBoost and SoFt is investigated. The features selected by SoFt and ranked high by wrapper-based CatBoost datasets are shown in Fig. 4(b). In the German dataset, SoFt selects only one feature in layer 0, and this feature is also selected as the top feature in the ten train-test splits of the wrapper-based CatBoost. As a result, SoFt and CatBoost have similar log loss. From Table 9, it can be seen that MECP selected a large number of categorical features for Australian and Bank datasets.

#### 3.2.3. Balanced and numerical (B,N)

From Table 7, it can be seen that For (B, N) datasets, the percentage reduction in the size of the feature set in SoFt is in the range of 80% to 92%, which is very high. Consequently, the log loss of SoFt is also very high compared to that of CatBoost with a full feature set. To be specific, it can be seen from Table 7 that percentage reduction in the log loss with full feature set is in



(a) Imbalanced datasets with mixture features



(b) Balanced datasets with mixture features

**Fig. 4.** The figure depicts three components: (1) features selected by SoFt (i.e., MECP); they are plotted along $X$-axis, (2) number of times each feature is selected (i.e., frequency) by CatBoost in K-fold cross-validation; it is indicated by height of each bar, (3) average rank of each feature in CatBoost; it is represented by color, with green (red) corresponds to the lowest (highest) rank; note that the ranks are stated in terms of percentile to account for different sizes of datasets (refer Section 3.2.1 for more details); green color of bars reflect the fact that features selected by MECP are consistently among the top-ranked features of CatBoost.

the range of 14% to 41%. It is also the case for the wrapper-based CatBoost as the feature subset size of wrapper-based CatBoost is constrained to be the same as SoFt.

In summary, for balanced datasets, the number of features selected by MECP are low that leads to higher log loss. A potential solution to resolve this issue is to select features from inner layers of Pareto front, i.e., features with large ranks from each feature score.

## 4. Conclusion

This work focuses on the design of robust yet simple and consistent predictive models for imbalanced and heterogeneous datasets. The robustness of two predictive models CatBoost and H2O GBM, with log loss as a performance metric, is evaluated to select the learning algorithm. CatBoost is shown to achieve lower log loss than H2O GBM for the majority of datasets. Moreover, the features selected by these algorithms are consistent, i.e., feature ranks are similar for both algorithms across all folds of a $K$-fold cross-validation. Users may lack the knowledge, resources, or time to tune the parameters of predictive models. In such scenarios, CatBoost with default parameters is shown to achieve better log loss than H2O GBM. The analysis in this work is limited to CatBoost and H20 GBM. The design of an optimal classifier for imbalanced and heterogeneous datasets is an interesting open research problem; we left it for the future work.

SoFt, which is a combination of CatBoost and a parameter-free filtering-based feature selection algorithm MECP, is shown to be robust and consistent. For imbalanced datasets, the log loss of SoFt is comparable with that of the CatBoost with a full feature set. On the other hand, the MECP selects too few features for the balanced datasets, leading to higher log loss for SoFt compared to the CatBoost with full feature set. If the number of features selected by wrapper-based CatBoost is constrained to be equal to the number of features selected by SoFt, the log loss of SoFt is shown to be comparable to the wrapper-based CatBoost. Moreover, most of the features selected by SoFt and ranked top by CatBoost are similar. The SoFt approach decouples the feature selection and learning algorithm. It eliminates the bias of the selected features to the learning algorithm. Therefore, the features generated by less computationally intensive SoFt can be used with an increased level of confidence for tasks such as the model-agnostic explanation of decisions taken by the learning algorithms. In summary, the proposed SoFt algorithm is the first step towards the automation of machine learning tasks.

## CRediT authorship contribution statement

**Gary Kee Khoon Lee:** Conceptualization, Methodology, Investigation, Writing – original draft, Review. **Henry Kasim:** Software, Investigation, Data curation, Validation, Formal analysis, Writing – review & editing. **Rajendra Prasad Sirigina:** Formal analysis, Writing – original draft, Writing – review & editing. **Shannon Shi Qi How:** Investigation, Validation, Data curation. **Stephen King:** Supervision, Writing – review & editing. **Terence Gih Guang Hung:** Supervision, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

The research undertaken in the Rolls-Royce@NTU Corporate Lab is supported by the Singapore Government.

## References

[1] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Syst. Appl. 73 (2017) 220–239.
[2] L. Wu, Y. Hu, X. Liu, X. Zhang, W. Chen, S. Alan, J.A. Kellum, L.R. Waitman, M. Liu, Feature ranking in predictive models for hospital-acquired acute kidney injury, Sci. Rep. 8 (1) (2018) 1–11.
[3] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Progr. Artif. Intell. 5 (4) (2016) 221–232.
[4] V. López, A. Fernández, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (7) (2012) 6585–6608.
[5] J. Frery, A. Habrard, M. Sebban, L. He-Guelton, Non-linear gradient boosting for class-imbalance learning, in: Second International Workshop on Learning with Imbalanced Domains: Theory and Applications, 2018, pp. 38–51.
[6] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Statist. (2001) 1189–1232.
[7] A.V. Dorogush, V. Ershov, A. Gulin, Catboost: gradient boosting with categorical features support, 2018, arXiv preprint arXiv:1810.11363.
[8] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[9] L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, in: Advances in Neural Information Processing Systems, 2018, pp. 6638–6648.
[10] M. Malohlava, A. Candel, Gradient Boosting Machine with H2O, H20 Booklet, 2017.
[11] M.R. Machado, S. Karray, I.T. de Sousa, Lightgbm: an effective decision tree gradient boosting method to predict customer loyalty in the finance industry, in: 2019 14th International Conference on Computer Science & Education, ICCSE, IEEE, 2019, pp. 1111–1116.
[12] I.T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments, Philos. Trans. R. Soc. A 374 (2065) (2016) 20150202.
[13] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Comput. Surv. 50 (6) (2017) 1–45.
[14] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (Mar) (2003) 1157–1182.
[15] H. Arai, C. Maung, K. Xu, H. Schweitzer, Unsupervised feature selection by heuristic search with provable bounds on suboptimality, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 666–672.
[16] G. Zames, N. Ajlouni, N. Ajlouni, N. Ajlouni, J. Holland, W. Hills, D. Goldberg, Genetic algorithms in search, optimization and machine learning, Inf. Technol. J. 3 (1) (1981) 301–302.
[17] P.-N. Tan, M. Steinbach, V. Kumar, Introduction To Data Mining, Pearson Education India, 2016.
[18] K. Grabczewski, N. Jankowski, Feature selection with decision tree criterion, in: Fifth International Conference on Hybrid Intelligent Systems, HIS'05, IEEE, 2005, p. 6.
[19] J. Ye, J.-H. Chow, J. Chen, Z. Zheng, Stochastic gradient boosted distributed decision trees, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, 2009, pp. 2061–2064.
[20] M. Robnik-Šikonja, I. Kononenko, Theoretical and empirical analysis of ReliefF and RReliefF, Mach. Learn. 53 (1–2) (2003) 23–69.
[21] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Advances in Neural Information Processing Systems, 2006, pp. 507–514.
[22] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 1151–1157.
[23] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, 2012.
[24] D.D. Lewis, Feature selection and feature extract ion for text categorization, in: Speech and Natural Language: Proceedings of a Workshop Held At Harriman, New York, February 23–26, 1992, 1992.
[25] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238.
[26] C.A. Mattson, A.A. Mullur, A. Messac, Smart Pareto filter: Obtaining a minimal representation of multiobjective design space, Eng. Optim. 36 (6) (2004) 721–740.
[27] E. Hancer, B. Xue, M. Zhang, D. Karaboga, B. Akay, A multi-objective artificial bee colony approach to feature selection using fuzzy mutual information, in: 2015 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2015, pp. 2420–2427.
[28] G. Enguerran, M. Abadi, O. Alata, An hybrid method for feature selection based on multiobjective optimization and mutual information, J. Inform. Math. Sci. 7 (1) (2015) 21–48.
[29] K. Neshatian, M. Zhang, Pareto front feature selection: using genetic programming to explore feature space, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, 2009, pp. 1027–1034.
[30] J. Jesus, A. Canuto, D. Araújo, Dynamic feature selection based on pareto front optimization, in: 2018 International Joint Conference on Neural Networks, IJCNN, IEEE, 2018, pp. 1–8.
[31] C. Molnar, Interpretable Machine Learning, Lulu. com, 2019.
[32] A. Adadi, M. Berrada, Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI), IEEE Access 6 (2018) 52138–52160.
[33] D. Gunning, Explainable Artificial Intelligence (XAI), Defense Advanced Research Projects Agency (DARPA), 2017, nd Web 2.
[34] CatBoost, Default parameters for CatBoost, 2020, URL https://catboost.ai/docs/concepts/python-reference_parameters-list.html. (Accessed 10 February 2020).
[35] H2O.ai, Default parameters for H2O GBM, 2020, URL http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm-faq/default_values.html. (Accessed 10 February 2020).
[36] J. Li, Scikit-feature, 2020, URL https://github.com/jundongl/scikit-feature. (Accessed 10 February 2020).
[37] A.J. Ferreira, M.A. Figueiredo, An unsupervised approach to feature discretization and selection, Pattern Recognit. 45 (9) (2012) 3048–3060.
[38] CatBoost, Description of the experimental setup, 2020, URL https://github.com/catboost/catboost/blob/master/catboost/benchmarks/quality_benchmarks/comparison_description.pdf. (Accessed 10 February 2020).
[39] H2O.ai, Tuning a GBM, 2020, URL http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm-faq/tuning_a_gbm.html. (Accessed 10 February 2020).
[40] N. Dmitriev, S. Tilga, N. Senderovich, A. Bezzubtseva, Benchmarks, GitHub, 2013, https://github.com/catboost/benchmarks/tree/master/quality_benchmarks.
[41] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, URL http://archive.ics.uci.edu/ml.
[42] KDD, KDD cup 2009 data challenge, 2009, URL https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Data.
[43] M. Christ, N. Braun, J. Neuffer, A.W. Kempa-Liehr, Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package), Neurocomputing 307 (2018) 72–77.
[44] M. Landry, Machine Learning with R and H2O, H2O. Ai, Mountain View, CA, USA, 2016.
[45] M. Cusumano-Towner, Boosting with log-loss, 2012, Link: https://pdfs.semanticscholar.org/1b3f/cf95f1f5450aa676bd6935851dd8dc121afd.pdf.
[46] W. Gao, Z.-H. Zhou, On the consistency of AUC pairwise optimization, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[47] Y.-W. Chen, C.-J. Lin, Combining SVMs with various feature selection strategies, in: Feature Extraction, Springer, 2006, pp. 315–324.