Sequential Datum–wise Feature Acquisition and Classifier Selection

Sachini Piyoni Ekanayake, *Student Member, IEEE*, Daphney-Stavroula Zois, *Member, IEEE*, Charalampos Chelmis, *Member, IEEE*

***Abstract*—We present a supervised machine learning framework for sequential datum–wise feature acquisition and classifier selection. The presented method sequentially acquires features during testing until it determines that additional features will not improve label assignment. At that stage, easy–to–classify examples are handled by a simple classifier, which assigns labels based on the lowest expected misclassification cost. On the contrary, difficult–to–classify examples are assigned a label using the acquired features along with one of a number of available complex classifiers. As more features are acquired, the presented framework continually assesses the difficulty of classifying each example. It controls both the feature acquisition and classifier selection processes through a carefully constructed optimization problem. We use eleven publicly available datasets to evaluate the presented framework with respect to accuracy and average number of acquired features, and obtain results when two and three complex classifiers are available, respectively. We compare the performance of the presented framework with both sequential feature aquisition methods and dynamic classifier selection methods, and observe improvements in accuracy as well as acquisition of less number of features on average. Moreover, we conduct experiments with popular ensemble classification methods and assess the performance of the proposed framework.**

***Impact Statement*—Traditional supervised classification typically relies on a single classifier that employs all of the available features to determine the classification outcomes of all examples in a dataset. However, in many real–world applications, features may not be free to acquire (e.g., gender information in a survey) or available at all (e.g., customer's transaction history for fraud detection). Similarly, the importance of features may differ across the entire dataset, while different classifiers may result in distinctively different classification outcomes. The framework proposed in this paper addresses the above challenges by sequentially acquiring features and selecting which classifier to use on a case–by–case basis. As a result, it improves accuracy by 50% while acquiring up to 84% less features on average compared to existing sequential feature acquisition methods. In addition, it achieves 2X better accuracy than existing dynamic classifier selection methods while acquiring up to 98% less number of features on average.**

***Index Terms*—supervised classification, instance–wise feature selection, inaccurate oracles, dynamic classifier selection, classifier pool**

Manuscript received (date to be filled by Editor). This material is based upon work supported by the National Science Foundation under Grants ECCS–1737443 & CNS–1942330.

S. P. Ekanayake and D.-S. Zois are with the Department of Electrical and Computer Engineering, University at Albany, State University of New York, Albany, NY (email: sekanayake@albany.edu, dzois@albany.edu).

C. Chelmis is with the Department of Computer Science, University at Albany, State University of New York, Albany, NY (email: cchelmis@albany.edu).

The Associate Editor coordinating the review of this manuscript and approving it for publication was (name to be filled by Editor).

I. Introduction

TRADITIONAL supervised classification adopts a batch–wise approach, where all features are readily available and used for determining the label of an example [1]. Unfortunately, in many real–world applications (e.g., medical and insurance assessments), this is not true since features are not freely available. For instance, in medical diagnosis, features come at a cost (i.e., due to the acquisition process), however, accurate classification is critical and time–sensitive. At the same time, the importance of features may differ among examples (e.g., in speech recognition, different individuals exhibit different accents, speech patterns and talk at different speeds). Considering all the above factors, making accurate predictions using the most informative features is an essential task in supervised classification. Consequently, feature acquisition, but more importantly, *instance–wise feature acquisition* has become an active research area [2]–[4].

On the other hand, *classifier selection* has also been an active area of research as using a single common classifier may not always be suitable for all examples in a dataset [5]–[7]. For instance, in image classification [5], different types of classifiers may perform better on different types of images due to factors such as image complexity and lighting. Alternatively, fusing multiple classifiers or considering the effect of all classifiers has been shown to increase accuracy, as seen in ensemble learning methods [1], [8], [9]. Nonetheless, the effectiveness of such approaches heavily depends on the ability to select the most appropriate classifier for each example, giving rise to *dynamic classifier selection* [6], [7], [10], [11]. For example, in [6], a decision model that employs dynamic selection of classifiers for the diagnosis of thyroid nodules is discussed, while in [7], dynamic selection of classifiers fused with a feature subspace clustering approach is used for analyzing insect bite hypersensitivity in horses using protein microarray data. Nevertheless, it should be noted that dynamic classifier selection approaches assume that all features are present at the time of classification, which may not hold true for several real–world applications.

To address the aforementioned issues, we propose a joint datum–wise feature aquistion and classifier selection framework. Our proposed framework acquires one feature at a time, and iteratively updates the posterior probability of the example belonging to each of the available classes. The resulting probability is used as a proxy to assess the difficulty of classifying the example under consideration, and controls the termination of the feature acquisition process. At that point,

the proposed framework decides which classifier to use by selecting among a simple classifier, based on the expected misclassification cost, and a set of complex but more powerful classifiers. Difficult–to–classify examples are forwarded to one of the later ones, while the remaining ones are handled by the simple classifier. In order to incorporate information about the cost associated with training/using each classifier, which may be available in real–world scenarios, our framework includes appropriate weighing parameters. We assess the performance of our proposed framework on eleven real–world datasets and compare it to eleven existing methods. We also conduct experiments with popular ensemble classification methods. Our results show that the proposed framework achieves a good balance between accuracy and average number of acquired features. Moreover, the inclusion of powerful classifiers improves accuracy by better handling difficult–to–classify examples while also saving on feature acquisition costs. Finally, we observe that low–cost classifiers are typically prioritized, while using them does not seem to significantly affect the overall performance.

The remaining sections are structured as follows. Section II provides a summary of pertinent prior research, while Section III presents the problem of sequential datum–wise feature acquisition and classifier selection followed by the optimum solution. Section IV presents detailed experiments that assess the performance of the proposed framework. Finally, Section V concludes the paper and briefly describes future research directions. All relevant code is available at https://github.com/IMAgINE9Lab/SFCS. Relevant proofs are included in the Appendix.

II. Literature Overview

In machine learning, *model selection* involves selecting the best model among a set of possible models to generalize well for unseen data [12]. This is typically accomplished using domain knowledge and/or model selection techniques such as cross–validation [1]. It is worthwhile to note that in the context of model selection, models vary based on the values of the hyperparameters [12]. Alternatively, instead of selecting one model, *ensemble learning* involves averaging multiple models, such as averaging predictions for regression or carrying out majority voting for classification [9], [13], [14]. Moreover, in the context of classification, *classifier fusion* methods combine multiple classifiers either at the output (e.g., via voting) or the classifier level (e.g., *dynamic classifier* methods) [15]. The framework presented in this paper is most related to the latter approach, which is discussed in more detail next.

Dynamic classifier methods, which involve selecting either a single classifier or a group of classifiers for each test example, have received considerable attention in machine learning [10]. Two types of dynamic selection methods exist: *Dynamic Ensemble Selection* (DES) and *Dynamic Classifier Selection* (DCS). DES involves selecting a set of classifiers and fusing their outputs to determine the final classification outcome for each test example. For instance, in [11], [16]–[18], majority voting is employed to classify each example. In contrast, and relevant to our work, DCS selects a *single classifier* based on the set of available classifiers and proceeds to generate a classification outcome. Specifically, a pool of classifiers is trained and the region of competence of each classifier in the pool is identified using techniques like k–NN and clustering. During testing, this information is used to help dertermine for each test example the region of competence and hence, the best–performing classifier to use. Among the various DCS methods, Overall Local Accuracy (OLA) [19], Local Class Accuracy (LCA) [19], and Modified Local Accuracy (MLA) [20] are considered state–of–the–art [21], and their main difference relies on the metric used to evaluate the performance of base classifiers in competence regions. The proposed framework differs from DCS in that it does *not* use all the available features to determine the classification outcome of each example. In fact, it sequentially acquires the features that seem more relevant to each example before determining the classification outcome. Additionally, it employs a single different classifier for each example, which is selected at the time of the termination of the feature acquisition process. This is in sharp contrast to DCS, which first uses all available classifiers to determine the classification outcome of each example during testing, and then carries out dynamic classifier selection as described above only for those examples where disagreements exists. Thus, the proposed framework reduces computational cost and is more practical for time–critical real–world applications.

Standard supervised classification methods (e.g., Support Vector Machines (SVM), Naive Bayes (NB)) consider all features available during training and testing. In contrast, offline feature selection (e.g., L1–norm based feature selection (Lasso)) selects a subset of features during training and uses it during testing. Recently, search space optimization (e.g., Enhanced Binary Butterfly Optimization Algorithm (BBOA) [22]) has been proposed to improve accuracy while using fewer features. In this case, *a subset of features* is first selected by combining different binary variants of BOA with Adaptive β–Hill Climbing. A standard classifier (e.g., SVM, NB) is then employed to evaluate performance of selected features. Due to their high performance, such methods are widely used in practice [22]–[25], however, they base their decisions on the same set of features irrespective of the example under consideration.

To accommodate prohibitively large feature spaces, *streaming or incremental feature selection* [26]–[28] selects features during training as these sequentially arrive one at a time. A new feature is added to the model if it most likely improves performance. The process terminates when a certain performance threshold is met or a specific number of features is selected. During testing, all examples are classified using the *same* selected subset of features. Static *instance–wise* approaches [29], [30] perform datum–wise classification, but access all features during testing for that purpose. Specifically, the approach described in [29] utilizes a neural network–based method inspired by the actor–critic model to identify a varying subset of features for each example. In [30], a method is proposed to reduce search space complexity in feature selection by using a threshold on the number of feature subsets. The approach uses a mixture of deep neural networks to determine the most relevant features for each example,

but it requires knowledge of all the feature values during testing. In contrast, *dynamic instance–wise* feature acquisition methods [3], [4], [31] adaptively acquire different features one at a time to classify each example during testing. Specifically, in [3], the problem of joint dynamic instance–wise feature acquisition and classification is introduced and solved. The properties of the optimum solution are studied, and two new algorithms are proposed. [4] extends [3] by modeling feature dependencies with a Bayesian network, which is then used to sequentially acquire the most informative features to classify each example. Finally, [31] studies the problem of dynamic instance–wise feature acquisition and classification when multiple classification variables are present and are related through a known Bayesian network. In this context, features are dynamically acquired for each variable and each example, while classification decisions are propagated through the Bayesian network to enhance overall performance. A major drawback of the latter approaches is inherently their classification mechanism; labels are assigned based on the smallest expected misclassification cost defined in terms of the posterior probability of the label of the example under consideration given the information provided by the already acquired features. Such mechanism may work well for some examples, but not for all, leading to considerable performance degradation. Moreover, in contrast to the above methods, which employ a single classifier to determine the classification outcome for each example, the proposed framework provides the flexibility to choose one out of a number of classifiers (i.e., simple or complex powerful classifiers). As a result, a distinct classifier may be used along with the distinct features acquired to classify each example.

III. Problem Description & Solution

In standard supervised classification, the intent is to learn a model that maps feature vector $X \triangleq [X_1, \ldots, X_F]^T$ to a label $Y = y \in \{1, \ldots, N\}$, where X_f represents a feature and the value of X is denoted as $x \triangleq [x_1, \ldots, x_F]^T$. The assumption is that the entire feature vector is accessible during both training and testing. Herein, we consider this problem under a slightly different context. Specifically, all features are available during training, but in testing, the features of each example are *sequentially* acquired one at a time based on a pre–defined fixed order (c.f. Fig 3). Further, acquiring feature X_f during *testing* incurs cost $c_f > 0, f = 1, \ldots, F$. Inherently, acquiring less features can save on acquisition costs. However, we may not have adequate information to make a reliable label assignment.

In this section, we describe the problem of sequential datum–wise joint feature acquisition and classification when multiple classifiers are present. The objective is to *jointly* acquire the *subset of features* based on which each example is to be classified, the appropriate *classifer* to be used for this task, and the respective *label assignment* for each example in the testing dataset. The benefit of having access to multiple classifiers is twofold. First, examples that require the acquisition of large number of features to be accurately classified could instead be potentially classified by a more powerful classifier using less features. Second, simple but less accurate classifiers may be inaccurate for certain examples regardless of the number of acquired features; such difficult–to–classify examples may be better handled by a more powerful classifier. Next, we define a number of variables needed to mathematically describe the problem of interest and present the optimization problem to be solved.

A. Optimization Problem

We define random variables S, U_S and D_S. $S \in \{0, \ldots, F\}$ is the *last feature* acquired before assigning a label to Y. $S = 0$ means no features have been acquired. $U_S \in \{0, \ldots, Z\}$ represents the *classifier* selected after S features have been acquired. Specifically, $U_S = 0$ represents the selection of a simple (possibly less accurate) classifier, while $U_S \in \{1, \ldots, Z\}$ represents the selection of one out of a set $C \triangleq \{C_1, \ldots, C_Z\}$ of more complex and powerful classifiers. For example, such classifiers could be Support Vector Machine (SVM) and Decision Tree (DT). Last but not least, we define $D_S \in \{1, \ldots, N\}$ as the label assignment provided by classifier U_S based on S acquired features. To learn a model that jointly selects the number of acquired features, the classifier to be used, and the label assignment for each example, we propose the following cost function:

$$L(S, U_S, D_S) = \mathbb{E}\Bigg\{ \sum_{f=1}^{S} c_f + \sum_{z=1}^{Z} \lambda_z \mathbb{I}_{\{U_S = z\}} h_S^z + \gamma \mathbb{I}_{\{U_S = 0\}} \sum_{j=1}^{N} \sum_{i=1}^{N} \Omega_{ij} P(D_S = j, Y = i) \Bigg\}, \quad (1)$$

where $\gamma \triangleq 1 - \sum_{z=1}^{Z} \lambda_z, \gamma > 0$ and $\{\lambda_z \in (0, 1)\}$ is a set of weighing parameters to differentiate between the various classifiers. Further, we define $\lambda \triangleq \{\lambda_0, \lambda_1, \ldots, \lambda_Z\}$, where $\lambda_0 \triangleq \gamma$ and $\lambda_t \in \lambda, t = 0, \ldots, Z$. Here $\Omega_{ij}, i, j \in \{1, \ldots, N\}$, is the cost of assigning label j to an example when the true label is i. The term $h_S^z, z = 1, \ldots, Z$, representing the cost associated with selecting to use classifier C_z when S features have been acquired, is defined as:

$$h_S^z \triangleq \sum_{i=1}^{N} P_z(e, Y = i | x_1, \ldots, x_S). \quad (2)$$

It is defined in terms of the error probability $P_z(e, Y = i | x_1, \ldots, x_S)$ of classifier C_z when the true label of an example is i and S features have been acquired. The first term in Eq. (1) denotes the cost of acquiring S features. The second term indicates the cost of selecting and using one out of the Z powerful classifiers to assign a label to an example using S acquired features. Finally, the last term captures the same cost but in the case where the simple (but possibly less accurate) classifier is selected. Therefore, our goal is to minimize the expected cost in Eq. (1) and determine the optimum values of S, U_S and D_S for each example in the testing dataset.

B. Posterior Probability

Let $\phi_f^i \triangleq P(Y = i | x_1, \ldots, x_f), f = 1, \ldots, F, i = 1, \ldots, N$, denote the *posterior* probability the label of an

example currently under consideration being i when f features have been so far acquired. We use Bayes' rule to update this *posterior* probability when a new feature is acquired, i.e.,

$$\phi_f^i = \frac{P(x_f|Y=i)\phi_{f-1}^i}{P(x_f|Y=1)\phi_{f-1}^1 + \ldots + P(x_f|Y=N)\phi_{f-1}^N}. \quad (3)$$

In this context, $\phi_f \triangleq [\phi_f^1, \ldots, \phi_f^N]^T, f = 1, \ldots, F$, represents the *posterior probability vector*, where $\phi_0 \triangleq [\rho_1, \ldots, \rho_N]^T$ denotes the case where no features have been acquired. Here, $\rho_i = P(Y=i), i = 1, \ldots, N$, represents the *prior* probability the label of the example currently under consideration being i. In line with prior work in the area [3], we assume features to be independent given the label of the example. We underscore that this assumption results in a good trade–off between accuracy and average number of acquired features per example, irrespective of its simplicity (c.f. Section IV).

C. Problem Solution

To minimize Eq. (1), we first use the probability vector ϕ_f to rewrite the cost of selecting between available classifiers in C (second term in Eq. (1)). Specifically, Corollary 1 expresses h_S^z in terms of ϕ_f (see Appendix A for proof).

Corollary 1: For given feature values $[x_1, \ldots, x_S]$, the cost h_S^z associated with selecting to use classifier C_z when S features have been acquired can be expressed as:

$$h_S^z = \mathbf{Q}_{S,z}^T \phi_S, \quad (4)$$

where $\mathbf{Q}_{S,z} \triangleq [Q_{S,z}^1, \ldots, Q_{S,z}^N]^T$ and $Q_{S,z}^i \triangleq P_z(e|Y = i, x_1, \ldots, x_S)$.

Corollary 1 enables us to express Eq. (1) as follows:

$$L(S, U_S, D_S) = \mathbb{E}\Bigg\{ \sum_{f=1}^{S} c_f + \sum_{z=1}^{Z} \lambda_z \mathbb{I}_{\{U_S=z\}} \mathbf{Q}_{S,z}^T \phi_S + \gamma \mathbb{I}_{\{U_S=0\}} \sum_{j=1}^{N} \sum_{i=1}^{N} \Omega_{ij} P(D_S = j, Y = i) \Bigg\}. \quad (5)$$

Eq. (5) is then further simplified to Eq. (6) using the definition of indicator function and the posterior probability as in prior work [3]. Specifically, $P(D_S = j, y = i) = \mathbb{E}\{\phi_f^i \mathbb{I}_{\{D_S=j\}}\}$ and defining $\mathbf{\Omega}_j \triangleq [\Omega_{1j}, \ldots, \Omega_{Nj}]^T$, we obtain:

$$L(S, U_S, D_S) = \mathbb{E}\Bigg\{ \sum_{f=1}^{S} c_f + \sum_{z=1}^{Z} \lambda_z \mathbb{I}_{\{U_S=z\}} \mathbf{Q}_{S,z}^T \phi_S + \gamma \mathbb{I}_{\{U_S=0\}} \sum_{j=1}^{N} \mathbf{\Omega}_j^T \phi_S \mathbb{I}_{\{D_S=j\}} \Bigg\}. \quad (6)$$

Next, we proceed with minimizing the expression in Eq. (6) in three steps. To this end, we first look into determining the optimum label assignment method D_S^* when fixed S features have been acquired and a classifier has been selected (i.e., U_S is fixed). In this context, depending on the selected value of U_S, there are two distinct cases. Specifically, if $U_S = 0$, then the optimum label assignment is carried out by the simple classifier, as discussed next. Consider the last term of Eq. (6). As discussed in prior work [3], for any D_S, $\gamma \sum_{j=1}^{N} \mathbf{\Omega}_j^T \phi_S \mathbb{I}_{\{D_S=j\}} \geq \gamma G(\phi_S)$ where $G(\phi_S) \triangleq \min_{1 \leq j \leq N} [\mathbf{\Omega}_j^T \phi_S]$. Therefore, the simple classifier carries out the optimum label assignment D_S^* as follows:

$$D_S^* = \underset{1 \leq j \leq N}{\operatorname{argmin}} [\mathbf{\Omega}_j^T \phi_S]. \quad (7)$$

On the other hand, if $U_S \in \{1, \ldots, Z\}$, the optimum label assignment is obtained using one out of the powerful classifiers in set C. In this case, the selected classifier (e.g., SVM) uses the acquired S features, as it deemed fit by its internal mechanism, to assign a label to the example under consideration.

We subsequently look into determining the optimum classifier selection method U_S^* when fixed S features have been acquired. As evident from Eq. (6), the last two terms effectively control the classifier selection process. To this end, we define $J(\phi_S) \triangleq \min_{0 \leq t \leq Z} [\lambda_t H_S^t(\phi_S)]$, where $\lambda_0 \triangleq \gamma$, $H_S^0(\phi_S) \triangleq G(\phi_S)$ and $H_S^z(\phi_S) \triangleq \mathbf{Q}_{S,z}^T \phi_S, z = 1, \ldots, Z$. Therefore, $\sum_{z=1}^{Z} \lambda_z \mathbb{I}_{\{U_S=z\}} \mathbf{Q}_{S,z}^T \phi_S + \gamma \mathbb{I}_{\{U_S=0\}} G(\phi_S) = \sum_{t=0}^{Z} \lambda_t H_S^t(\phi_S) \mathbb{I}_{\{U_S=t\}}$. Since $\sum_{t=0}^{Z} \mathbb{I}_{\{U_S=t\}} = 1$ for any U_S, $\sum_{t=0}^{Z} \lambda_t H_S^t(\phi_S) \mathbb{I}_{\{U_S=t\}} \geq \sum_{t=0}^{Z} J(\phi_S) \mathbb{I}_{\{U_S=t\}}$. The right side of the inequality is independent of U_S because $J(\phi_S)$ is the minimum from a set of numbers. Then the inequality simplifies to $\sum_{t=0}^{Z} \lambda_t H_S^t(\phi_S) \mathbb{I}_{\{U_S=t\}} \geq J(\phi_S)$. Thus, the optimum classifier selection method U_S^* is given by:

$$U_S^* = \underset{0 \leqslant t \leqslant Z}{\operatorname{argmin}} [\lambda_t H_S^t(\phi_S)]. \quad (8)$$

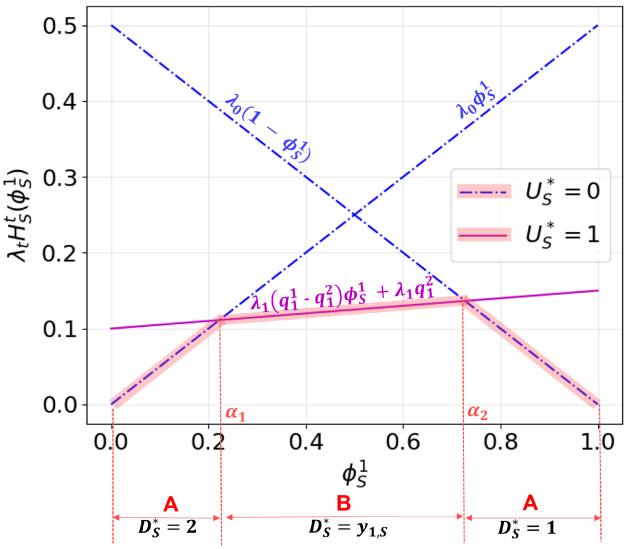


Fig. 1: Illustration of the classifier selection and label assignment processes in the case of two label values (i.e., $N = 2$), a simple classifier (region A), and a single powerful classifier (region B). The highlighted line graphically represents Eq. (8).

Before proceeding with the last and final step of our derivations, we present an example to showcase the operation of the optimal classifier selection and label assignment methods. In particular, Fig. 1 illustrates how decisions are made based on the value of the posterior probability in the case of two label values (i.e., $N = 2$) and a single powerful classifier (i.e., $Z = 1, C = \{C_1\}$) in addition to the simple classifier. Further, the highlighted line graphically represents

Eq. (8) in this case. Details about the setting presented in Fig. 1 and relevant derivations are provided in Appendix B. We observe that depending on the value of the posterior probability ϕ_S^1, a different classifier selection choice is made. Specifically, region A represents selecting to use the simple classifier, while region B denotes selecting to use the powerful classifier C_1. The optimum label assignment for each region is also provided, where $y_{1,S}$ represents the label assignment from powerful classifier C_1 using S number of features i.e., $y_{z,S} \triangleq M_z(X_1, \ldots, X_S)$ where $M_z(.)$ is the trained model of C_z (c.f. Section III-D). The operation of the optimal classifier selection and label assignment methods is very intuitive, as explained next. When the posterior probability ϕ_S^1 (i.e., the probability of the label of the example being 1) takes relatively low values (e.g., below around 0.2), the simple classifier is selected and the example under consideration is assigned label 2. In this case, the framework is confident about the label assignment and prefers to select the simple classifier. Similar is the case where the posterior probability ϕ_S^1 is relatively high (e.g., above around 0.7). In contrast, when the posterior probability ϕ_S^1 falls in the intermediate interval, the framework is not very confident about the true label of the example under consideration. In that case, the powerful classifier is preferred that can potentially lead to an accurate label assignment. As we will later see (c.f. Section III-D), when the posterior probability falls in that region, the proposed framework tends to acquire more features in the hope that the resulting posterior probability will end up in region A. If this is not the case, the powerful classifier will still be selected and used in the end. In summary, difficult–to–classify examples (as indicated by the value of the posterior probability) are essentially forwarded to classifier C_1, while the rest are handled by the simple classifier.

Finally, we look into determining the optimum feature acquisition method S^*. Since we have already determined the optimal classifier selection and label assignment methods, the cost function in Eq. (6) now becomes:

$$\tilde{L}(S) = \mathbb{E}\left\{ \sum_{f=1}^{F} c_f + l(\phi_S) \right\}, \tag{9}$$

where $l(\phi_S) \triangleq \min_{0 \leqslant t \leqslant Z} [\lambda_t H_S^t(\phi_S)]$. The form of the cost function in Eq. (9) enables us to use stochastic dynamic programming [32] to determine the optimum feature acquisition method. In particular, we obtain the following dynamic programming equation:

$$\bar{L}_f(\phi_f) = \min \left[l(\phi_f), \bar{I}_f(\phi_f) \right], \tag{10}$$

where

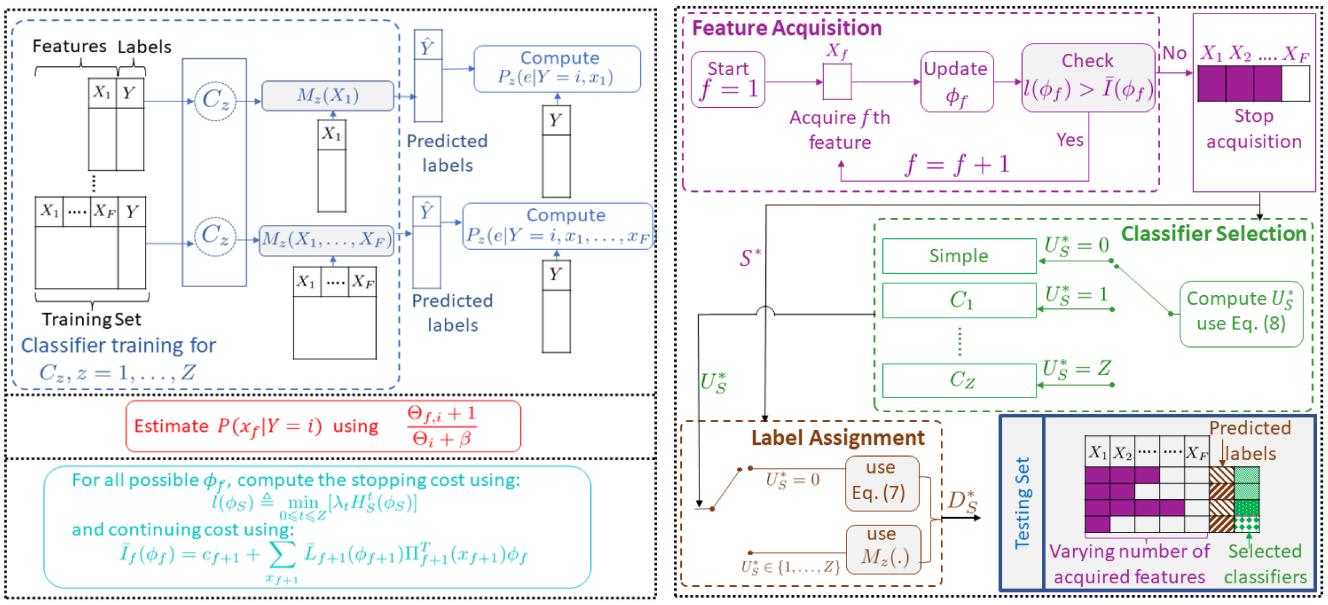
$$\bar{I}_f(\phi_f) = c_{f+1} + \sum_{x_{f+1}} \bar{L}_{f+1}(\phi_{f+1}) \Pi_{f+1}^T(x_{f+1}) \phi_f \tag{11}$$

with $\bar{L}_F(\phi_F) = l(\phi_F)$ and $\Pi_f(x_f) \triangleq [P(x_f|Y = 1), \ldots, P(x_f|Y = N)]^T$. $\bar{I}_f(\phi_f)$ represents the cost of continuing feature acquisition, while $l(\phi_f)$ represents the cost of stopping this process. Therefore, if the former term is less than the latter, the optimum feature acquisition method will keep acquiring features. Otherwise, it will stop and proceed with selecting one out of the available classifiers to decide on a label assignment. In the end, the optimum number of acquired features will differ for each example under consideration, and be either $S^* = f < F$, or $S^* = F$ if all features are acquired.

D. Proposed Method

Our proposed method (see Fig. 2) involves two phases: training (Fig. 2a) and testing (Fig. 2b), which follow the problem solution described in Section III-C. During training, all possible posterior probability vectors ϕ_f are generated by discretizing the range $[0, 1]$ such that $\phi_f \mathbf{1}^T = 1$. Here, $\mathbf{1}$ is a N–dimensional vector of all ones. Specifically, considering the arithmetic precision of discretization to be η, d possible vectors ϕ_f are generated. Then, for each of ϕ_f, Eqs. (8) and (10) are numerically solved to determine the optimum feature acquisition and classifier selection processes. Moreover, classifiers $C_z, z = 1, \ldots, Z$, are trained for each number $f = 1, \ldots, F$, of features (see Fig. 2a) and the conditional probabilities $P(x_f|Y = i)$ and $P_z(e|Y = i, x_1, \ldots, x_S)$ are estimated (c.f. Section IV).

Next, we conduct the complexity analysis of the training phase. First, computing $\bar{I}_f(\phi_f)$ involves computational complexity of $\mathcal{O}(FN\beta\eta^{N-1})$ [3]. Here, β represents the number of bins used to discretize the feature space (c.f. Section IV-B). Furthermore, computing $l(\phi_f)$ involves evaluating Eq. (8). In particular, computing the dot product of N terms and finding the minimum is $\mathcal{O}(N^2)$. Therefore, computing $\lambda_0 H_f^0(\phi_f)$ for all possible d vectors ϕ_f is $\mathcal{O}(N^2\eta^{N-1})$ [3]. Computing $\lambda_z H_f^z(\phi_f), z = 1, \ldots, Z$, however, is $\mathcal{O}(ZN^2\eta^{N-1})$, because we repeat this process for all powerful classifiers (in total Z). Thus, the computational complexity to numerically solve Eqs. (8) and (10) is $\mathcal{O}((F\beta + ZN)N\eta^{N-1})$. The former term can be simplified to $\mathcal{O}(F\beta N\eta^{N-1})$ if the number of classes and the available classifiers is low. Second, during training, all powerful classifiers $C_z, z = 1, \ldots, Z$, are trained for all f number of features, where $f = 1, \ldots, F$. The overall complexity of this step is dictated by the cost of training the most expensive of all classifiers. Namely, this is $\mathcal{O}(\sum_{f=1}^{F} \Xi^{\text{train}}(f, N, \Theta_{\text{train}}))$, where $\Xi^{\text{train}}(f, N, \Theta_{\text{train}})$ denotes the cost of training the most expensive classifier when f features are used and Θ_{train} is the total number of examples in the training dataset. Furthermore, to estimate error probability $P_z(e|Y = i, x_1, \ldots, x_S)$ involves using the learned model $M_z(.)$ to obtain a label assignment for each example in the training dataset. Since there are Z available classifiers and F possible features, this step requires $\mathcal{O}(\sum_{f=1}^{F} \Xi^{\text{test}}(f, N, \Theta_{\text{train}}) + FZ\Theta_{\text{train}})$. In the former expression, $\Xi^{\text{test}}(f, N, \Theta_{\text{train}})$ is the testing complexity of the most expensive classifier when f features are used. Finally, estimating $P(x_f|Y = i), i = 1, \ldots, N$, requires $\mathcal{O}(\Theta_{\text{train}})$. Therefore, the overall complexity of training all powerful classifiers and estimating relevant parameters is $\mathcal{O}(\Xi^{\text{total}}(F, N, \Theta_{\text{train}}) + FZ\Theta_{\text{train}})$, where $\Xi^{\text{total}}(F, N, \Theta_{\text{train}}) \triangleq \sum_{f=1}^{F} \left(\Xi^{\text{train}}(f, N, \Theta_{\text{train}}) + \Xi^{\text{test}}(f, N, \Theta_{\text{train}}) \right)$. As an example, consider the case of $Z = 1$ classifier, i.e., binary SVM with training complexity $\mathcal{O}(\Theta_{\text{train}}^2 F)$ [33] and testing complexity $\mathrm{O}(\nu F)$ [34], where ν represents the number of support vectors. Adopting an one–vs–rest approach for multiclass classification for SVM, the overall training complexity



(a) (b)

Fig. 2: Graphical representation of the proposed approach. (a) Training phase: training of powerful classifiers $C_z, z = 1, \ldots, Z$, parameter estimation, and computing numerical solutions, (b) testing phase: feature acquisition, classifier selection, and label assignment for a single example. The final output with acquired features, predicted labels, and selected classifiers for all the examples in the testing dataset is also presented.

becomes $\mathcal{O}(FN\beta\eta^{N-1} + N^2\eta^{N-1} + N\Theta_{\text{train}}^2 F)$. Now, consider two powerful classifiers, i.e., binary SVM and binary DT. The training and testing complexities of binary DT are $\mathcal{O}(F\Theta_{\text{train}} \log(\Theta_{\text{train}}))$ and $\mathcal{O}(\log(\Theta_{\text{train}}))$[1], respectively [33]. Thus, the overall training complexity remains the same, i.e., $\mathcal{O}(FN\beta\eta^{N-1} + N^2\eta^{N-1} + N\Theta_{\text{train}}^2 F)$, since the most expensive classifier to train is SVM.

During testing, numerical solutions obtained during training are utilized to carry out feature and classifier selection as well as label assignment. First, we initialize the posterior probability $\phi_0 \triangleq [\rho_1, \ldots, \rho_N]^T$, where $\rho_i = P(Y = i), i = 1, \ldots, N$ (c.f. Section IV-B). Then, features are sequentially acquired based on Eq. (10). Specifically, at each stage, a new feature is acquired if the stopping cost is greater than the continuing cost (see Fig. 2b), both of which have been computed during training. The process continues until a subset of features is acquired, or no more features are available. Eq. (8) is then used to decide which classifier to use (see Classifier Selection block in Fig. 2b) when the feature acquisition process terminates. At that point, Eq. (7) is used by the simple classifier to assign a label to the example currently under consideration. On the other hand, if one out of the powerful classifiers is selected, the acquired features are forwarded to that classifier to be used by $M_z(.)$ for label assignment. Algorithm 1 outlines the testing process of our proposed method, referred to as Sequential Datum–wise Feature Acquisition and Classifier Selection (SFCS).

Next, we conduct the complexity analysis of Algorithm 1. At each stage f, SFCS acquires a single feature in $\mathcal{O}(1)$, and then updates the posterior probability vector. The latter step involves computing N terms requiring $\mathcal{O}(N)$ complexity. Thus, the overall time complexity of the feature acquisition process across all stages is $\mathcal{O}(FN)$. At the same time, comparing $l(\phi_f)$ and $\bar{I}_f(\phi_f)$ is $\mathcal{O}(1)$. To determine which classifier to use when the feature acquisition process terminates, we need to compute $Z + 1$ terms (see Eq. (8)) and determine

[1]Note that the training dataset is used to estimate the error probabilities.

Algorithm 1 SFCS method

Input: test example $[x_1, \ldots, x_F]$, numerical solutions $\{l(\phi_f), \bar{I}_f(\phi_f)\}$, set $\{M_1(.), \ldots, M_Z(.)\}$ of trained powerful classifier models
Output: classification decision Val_Y
Initialize $\phi_0 \triangleq [\rho_1, \ldots, \rho_N]^T$
for each feature $f \in F$ **do**
  Acquire feature x_f
  Update ϕ_f using Eq. (3)
  if $l(\phi_f) < \bar{I}_f(\phi_f)$ **then**
    Use Eq.(8) to determine which classifier to use
    if $U_S^* = 0$ **then**
      Determine label assignment D_S^*
      Set $Val_Y = D_S^*$
    else if $U_S^* \in \{1, \ldots, Z\}$ **then**
      Set $Val_Y = y_{z,S}$
    end if
    Terminate feature acquisition process
  else
    Continue feature acquisition process
  end if
end for
Return: Val_Y

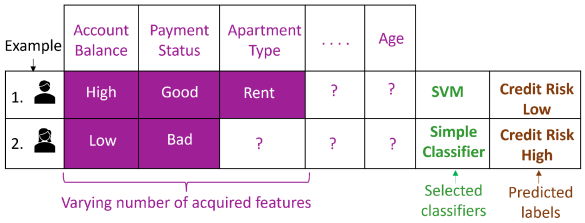


Fig. 3: Illustration of SFCS for 2 examples in the CREDIT dataset.

TABLE I: Datasets & their characteristics (number of examples (Θ), number of features (F), number of classes (N)).

Dataset	Θ	F	N
Monks	601	6	2
Diabetes	768	8	2
Eye	14980	14	2
Magic	19020	10	2
Student	649	32	4
Credit	1000	20	2
Travel	5454	23	5
Wine	178	12	3
Gender	4746	20	3
Spambase	4601	57	2
Madelon	2600	500	2

the minimum among them. Specifically, computing $\gamma G(\phi_f)$ requires $\mathcal{O}(N^2)$ due to taking the inner product and identifying the minimum among N terms. Similarly, computing $\lambda_z H_S^z(\phi_S), z = 1, \ldots, Z$, requires $\mathcal{O}(ZN^2)$ considering that there are Z classifiers. Overall, solving Eq. (8) results in $\mathcal{O}(ZN^2)$. If the simple classifier is selected, determining the label of a single example is $\mathcal{O}(N^2)$. This consists of computing the dot product of N terms i.e., $\mathcal{O}(N^2)$, and finding the minimum $\mathcal{O}(N)$ (see Eq. (7)). In the opposite case, obtaining a label for the example under consideration depends on the selected classifier. As an example, considering that the testing complexity of binary SVM is $\mathcal{O}(\nu F)$ [33], [34], the overall time complexity of assigning a label to a single example is $\mathcal{O}(N^2 + \nu NF)$.

Fig. 3 illustrates the testing phase of SFCS (i.e., Fig. 2b) for 2 examples in the CREDIT dataset. Specifically, SFCS acquires 3 features (i.e., "Account Balance", "Payment Status", "Apartment Type") for example 1 and uses SVM to predict that the credit risk is low. In contrast, SFCS acquires 2 features (i.e., "Account Balance", "Payment Status") for example 2 and uses the simple classifier to predict that the credit risk is high.

IV. EXPERIMENTAL RESULTS

In this section, we present and discuss numerical experiments conducted to assess the performance of SFCS. The experiments are carried out on an Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz computer with 16 GB memory. All experimental results are five–fold cross-validated. Cross–validation is a standard technique used to assess the effectiveness of machine learning models and prevent overfitting. We use accuracy, the average number of acquired features, and Receiver Operating Characteristic (ROC) / Area Under the Curve (AUC) as evaluation metrics. Training and testing times (in seconds) are also reported.

A. Datasets

We use 11 real–world datasets to carry out experiments, and their characteristics are reported in Table I. MONKS PROBLEM (referred to MONKS in Table I), DIABETES, EEG EYE STATE (referred to EYE in Table I), MAGICTELESCOPE (referred to MAGIC in Table I), STUDENT PERFORMANCE (referred to STUDENT in Table I), WINE, SPAMBASE, and MADELON datasets are from OpenML [35]. GERMAN CREDIT (referred to CREDIT in Table I) and TRAVEL REVIEW RATINGS (referred to TRAVEL in Table I) datasets are from Kaggle [36], while GENDER GAP IN SPANISH (referred to GENDER in Table I) is from UCI machine learning repository [37]. All datasets are used as is, except for the STUDENT and TRAVEL datasets, as described next. In particular, we quantize the variable G_3 that represents the final grade in the STUDENT dataset such that it takes 4 values, i.e., $G_3 \in \{0, 1, 2, 3\}$[2]. In the TRAVEL dataset, the attribute *Average ratings on restaurants* is selected as the label Y taking five distinct values, i.e., $Y \in \{0, 1, 2, 3, 4\}$[3].

B. Experimental Setting

To numerically determine the optimum solution, the conditional probabilities $P(x_f|Y = i), f = 1, \ldots, F, i = 1, \ldots, N$, are needed. These are estimated as $\hat{P}(x_f|Y = i) = \frac{\Theta_{f,i}+1}{\Theta_i+\beta}$ during training. The notation $\Theta_{f,i}$ represents the count of examples with label i and x_f being a specific value, whereas Θ_i corresponds to the total number of examples with label i. The parameter $\beta \in \{2, 3, 5, 10, 20, 30, 40, 50, 100\}$ represents the number of bins considered, and we define the prior probability $P(Y = i) = 1/N$ to reflect the assumption of equiprobable labels. During training/testing and before any other processing is carried out, features are sorted in descending order of feature variance scaled by the cost coefficient. This avoids considering all exponentially large numbers of orders that features may have. This order promotes the acquisition of informative but low–cost features [3]. We consider two variations of SFCS that incorporate 2 and 3 powerful classifiers, respectively, namely SFCS–2X with $C \triangleq \{\text{NB, SVM}\}$, and SFCS–3X with $C \triangleq \{\text{NB, SVM, DT}\}$. We also consider another variation of SFCS (i.e., SFCS–2E with $C \triangleq \{\text{XGB, RF}\}$) that incorporates 2 ensemble learning methods, i.e., XG Boosting (XGB), and Random Forest with tree depths 10. During training, the error $P_z(e|Y = i, x_1, \ldots, x_S)$ for each classifier $C_z \in C$ is estimated. Specifically, trained models $M_z(X_1, \ldots, X_S), z = 1, \ldots, Z$, are used to predict labels, which are then compared with the true labels (see Section. III-D). Then the error probability is estimated as $\hat{P}_z(e|Y = i, x_1, \ldots, x_S) = \frac{\Theta_{e,i}}{\Theta_i}$. Here, $\Theta_{e,i}$ represents the count of error e for the label i, whereas Θ_i is the total number of examples with label i. When no features are acquired (i.e., $S = 0$), we set such probability

[2]Label 0 is assigned if $0 \leqslant G_3 < 9$, label 1 if $9 \leqslant G_3 < 12$, label 2 if $12 \leqslant G_3 < 15$, and label 3 if $15 \leqslant G_3 \leqslant 20$.

[3]Label 0 is assigned if rating $\in [0, 1]$, label 1 if rating $\in (1, 2]$, label 2 if rating $\in (2, 3]$, label 3 if rating $\in (3, 4]$, and label 4 if rating $\in (4, 5]$.

TABLE II: Accuracy ("Acc"), average number of acquired features ("Feat"), training ("Train") and testing ("Test") time in seconds for the proposed approach (SFCS–2X, SFCS–3X) and baselines. Acc values that are the highest and second highest are indicated as bold and gray–shaded, and gray–shaded, respectively. Feat values that are the smallest and second smallest are indicated as bold and gray–shaded, and gray-shaded, respectively.

Dataset	Metric	SFCS–2X	SFCS–3X	ETANA	NB	SVM	DT	Lasso	OLA–2X	OLA–3X	LCA–2X	LCA–3X	MLA–2X	MLA–3X
Monks	Acc	0.657	0.779	0.529	0.591	0.657	**0.922**	0.654	0.674	0.797	0.662	0.649	0.661	0.649
	Feat	**4.449**	5.234	5.188	6.000	6.000	6.000	4.800	6.000	6.000	6.000	6.000	6.000	6.000
	Train	0.074	0.143	0.029	0.002	0.017	0.002	0.006	0.008	0.008	0.009	0.005	0.011	0.013
	Test	0.049	0.057	0.030	0.001	0.007	0.001	0.002	0.007	0.007	0.007	0.008	0.006	0.007
Diabetes	Acc	0.759	0.730	0.749	0.751	0.674	0.706	**0.766**	0.758	0.750	0.758	0.757	0.759	0.755
	Feat	6.301	6.059	**5.935**	8.000	8.000	8.000	8.000	8.000	8.000	8.000	8.000	8.000	8.000
	Train	0.114	0.294	0.029	0.001	0.004	0.002	0.005	0.014	0.015	0.014	0.015	0.013	0.012
	Test	0.057	0.077	0.021	0.002	0.003	0.001	0.003	0.007	0.007	0.008	0.008	0.007	0.008
Eye	Acc	0.574	**0.751**	0.500	0.437	0.551	0.475	0.551	0.368	0.419	0.371	0.375	0.371	0.375
	Feat	**4.258**	8.109	12.261	14.000	14.000	14.000	13.400	14.000	14.000	14.000	14.000	14.000	14.000
	Train	61.092	48.510	1.316	0.004	4.774	0.106	0.496	5.346	5.701	5.364	5.676	6.922	6.452
	Test	1.787	1.030	0.043	0.001	0.776	0.006	0.009	0.315	0.353	0.309	0.334	0.421	0.467
Magic	Acc	0.808	0.816	0.775	0.727	0.806	0.819	0.789	**0.860**	0.856	0.823	0.823	0.824	0.823
	Feat	**5.946**	7.527	6.302	10.000	10.000	10.000	9.000	10.000	10.000	10.000	10.000	10.000	10.000
	Train	39.884	36.798	0.956	0.003	4.256	0.215	0.501	4.804	4.854	4.851	4.852	5.878	5.929
	Test	2.100	1.515	0.033	0.002	0.608	0.002	0.090	0.258	0.246	0.285	0.264	0.306	0.310
Student	Acc	0.835	**0.847**	0.790	0.532	0.592	0.801	0.650	0.639	0.676	0.592	0.592	0.592	0.590
	Feat	6.903	**4.772**	9.035	32.000	32.000	32.000	29.600	32.000	32.000	32.000	32.000	32.000	32.000
	Train	5.289	11.712	2.543	0.002	0.018	0.003	0.101	0.033	0.030	0.033	0.028	0.051	0.037
	Test	0.050	0.036	0.060	0.001	0.001	0.001	0.050	0.009	0.010	0.008	0.012	0.017	0.009
Credit	Acc	**0.761**	0.741	0.714	0.700	0.700	0.664	0.734	0.720	0.697	0.713	0.698	0.704	0.698
	Feat	12.088	**10.011**	11.846	20.000	20.000	20.000	17.800	20.000	20.000	20.000	20.000	20.000	20.000
	Train	0.568	0.693	0.140	0.002	0.031	0.006	0.017	0.032	0.028	0.032	0.029	0.035	0.028
	Test	0.122	0.112	0.082	0.001	0.005	0.002	0.004	0.009	0.010	0.010	0.006	0.010	0.010
Travel	Acc	**0.814**	0.781	0.676	0.615	0.710	0.733	0.660	0.774	0.779	0.759	0.765	0.759	0.765
	Feat	8.613	**8.477**	9.985	23.000	23.000	23.000	23.000	23.000	23.000	23.000	23.000	23.000	23.000
	Train	51.910	71.475	11.140	0.002	0.536	0.044	0.776	0.764	0.670	0.762	0.671	0.788	0.837
	Test	0.674	0.744	0.738	0.002	0.109	0.003	0.005	0.069	0.063	0.070	0.069	0.074	0.077
Wine	Acc	0.955	0.943	0.950	0.950	0.669	0.910	0.944	**0.983**	**0.983**	0.972	0.972	0.972	0.972
	Feat	5.982	6.442	**4.349**	13.000	13.000	13.000	8.200	13.000	13.000	13.000	13.000	13.000	13.000
	Train	0.791	0.989	0.266	0.001	0.003	0.002	0.028	0.003	0.004	0.004	0.006	0.004	0.006
	Test	0.010	0.010	0.008	0.001	0.001	0.001	0.009	0.004	0.010	0.004	0.006	0.004	0.006
Gender	Acc	**1.000**	**1.000**	0.965	0.588	0.588	**1.000**	0.928	0.961	0.954	0.954	0.955	0.954	0.955
	Feat	**1.884**	2.086	3.678	20.000	20.000	20.000	17.800	20.000	20.000	20.000	20.000	20.000	20.000
	Train	7.686	8.262	0.367	0.001	0.488	0.010	0.974	0.404	0.411	0.403	0.422	0.373	0.445
	Test	0.186	0.152	0.164	0.001	0.089	0.003	0.031	0.027	0.025	0.026	0.029	0.025	0.028
Spambase	Acc	0.880	0.903	0.835	0.826	0.690	0.886	0.909	0.916	0.915	**0.918**	0.917	**0.918**	0.917
	Feat	10.498	**8.222**	30.922	57.000	57.000	57.000	50.600	57.000	57.000	57.000	57.000	57.000	57.000
	Train	27.167	30.860	0.199	0.004	0.773	0.063	0.034	0.670	0.555	0.658	0.565	0.638	0.659
	Test	0.368	0.334	1.174	0.002	0.145	0.010	0.001	0.047	0.050	0.049	0.048	0.053	0.060
Madelon	Acc	0.698	0.708	0.621	0.593	0.617	**0.743**	0.560	0.590	0.653	0.587	0.643	0.587	0.643
	Feat	12.620	**10.900**	68.017	500.000	500.000	500.000	492.000	500.000	500.000	500.000	500.000	500.000	500.000
	Train	196.147	304.115	1.332	0.018	2.218	0.470	3.022	3.689	3.838	3.740	3.809	3.440	3.506
	Test	0.291	0.207	1.162	0.001	0.512	0.012	0.002	0.319	0.358	0.325	0.335	0.301	0.317
Rank	Acc	**4.32**	4.59	8.23	10.59	9.86	6.77	7.86	5.32	5.59	6.77	7.09	6.77	7.23
	Feat	**1.82**	1.91	2.45	8.91	8.91	8.91	4.64	8.91	8.91	8.91	8.91	8.91	8.91

to $1/N$ assuming a random choice between the N labels. We consider $\Omega_{ij} = 1, \forall i \neq j$ and $\Omega_{ii} = 0, \forall i, j \in \{1, \ldots, N\}$. We assume feature cost to be the same for all features, i.e., $c_f = c, \forall f = 1, \ldots, F$.

C. Results & Discussion

We compare SFCS–2X and SFCS–3X with: (i) instance–wise joint feature selection and classification algorithm ETANA [3], (ii) the *offline* feature selection algorithm L1–norm based feature selection (Lasso), (iii) supervised learning algorithms NB, SVM with Gaussian kernel, and Decision Tree (DT), and (iv) dynamic classifier selection algorithms, OLA, LCA, and MLA [21]. We consider the same pool of classifiers as SFCS for the DCS algorithms, resulting in the following variations: OLA–2X, OLA–3X, LCA–2X, LCA–3X, MLA–2X, and MLA–3X. The DESlib [21] is used for DCS methods with default parameter values (e.g., $k = 7$) following the literature [10], [16], [18]. The values of $\eta = 10, V = 10$, and $e = 0.0001$ [3] are considered for ETANA. For SFCS, a grid search was conducted over the possible values of hyperparameters λ, c, and β. The results that yield the best accuracy using the smallest average number of features are

presented in Table II. In all the experiments, $\eta = 10$.

SFCS outperforms, or is competitive with, all baselines in terms of accuracy and the average number of acquired features. Moreover, it is evident that SFCS consistently performs better irrespective of the external classifier pool. Specifically, for all the datasets except DIABETES, CREDIT, and WINE, the accuracy increase of SFCS–3X is between 0.02% and 30.93% when compared to SFCS–2X. This improvement, however, is achieved by utilizing 10.72% to 90.43% more features on average except for STUDENT, TRAVEL, SPAMBASE and MADELON datasets. It is intuitive that fusing with a powerful external classifier such as DT, which performs better than SVM and NB, can lead to accuracy improvements. In the EYE dataset, DT is second to SVM in accuracy, but all the external classifiers have lower accuracy, nearly around 0.5. In this case, to achieve better accuracy, SFCS–3X utilizes 90.43% more features on average compared to SFCS–2X.

Comparing SFCS with ETANA, a recently proposed instance–wise joint feature selection and classification algorithm, it is observed that SFCS–2X achieves better accuracy (0.52% to 24.23%) in all datasets, while SFCS–3X's accuracy is improved with respect to ETANA from 3.60% to 50.20%, except for the DIABETES and WINE datasets. Additionally, the number of average acquired features in these cases is smaller (5.65% to 81.45% in SFCS–2X, and 15.10% to 83.97% in SFCS–3X) compared to ETANA, except for the DIABETES, CREDIT, and WINE datasets for SFCS–2X, and MONKS, DIABETES, MAGIC, and WINE for SFCS–3X. In the DIABETES and WINE datasets, the accuracy of SFCS–3X is only 2.45% and 0.65% less than ETANA, respectively, while acquiring a few more features. The decrease in accuracy may be due to the lower performance of DT, which is observed when DT is used as a standalone classifier for these two datasets. On the other hand, for the MONKS and MAGIC datasets, the accuracy improves (47.17% and 5.27%, respectively) compared to ETANA by using 0.90% and 19.44% more features, respectively. These observations validate that the addition of powerful classifiers can indeed enhance accuracy, mainly when these classifiers are utilized to assign labels to certain examples.

We observe that SFCS outperforms offline Lasso in almost all datasets except DIABETES and SPAMBASE. SFCS achieves better accuracy (between 0.51% to 36.32%) using 7.31% to 97.78% fewer features on average, with the exception of MONKS (9.04% more features for SFCS–3X). For the DIABETES dataset, there is a small accuracy drop (0.85% for SFCS–2X and 4.60% for SFCS–3X compared to Lasso), yet fewer number of features on average are acquired (21.24% and 24.27% fewer features for SFCS–2X and SFCS–3X). For SPAMBASE, there is also a small accuracy drop (3.20% and 0.72% for SFCS–2X and SFCS–3X) but fewer average number of features (79.25% and 83.75% for SFCS–2X and SFCS–3X) are acquired compared to Lasso.

After comparing SFCS with the DCS algorithms, it is evident that although DCS methods outperform the rest of the baselines, SFCS in turn outperforms DCS methods in terms of accuracy and, more importantly, average number of acquired features; in over half of the datasets, SFCS displays better accuracy (between 1% to 100%) compared to DCS methods, while acquiring 13% to 98% fewer features on average. In cases where DCS methods perform better in accuracy, SFCS–2X and SFCS–3X acquire, on average, fewer features (between 21.24% and 85.58%).

Finally, we compute Gini Impurity Reduction (GIR) [38], an extension of the Gini score [39], to measure the feature significance for SFCS and the baselines. Specifically, a feature with higher GIR is more significant than a feature with lower GIR, since the latter cannot be used to effectively separate the labels. SFCS and ETANA acquire different number of features per example. Thus, to compute GIR in this case, we began by separating examples in subsets according to the features acquired and used for their classification. For each such subset, we calculated the difference (i.e., reduction) between the Gini impurity of the label variable, and the weighted average of the Gini impurity of each feature. We subsequently computed the average GIR over the number of features and examples in a subset, and averaged over five–folds. In Fig. 4, the distribution of average GIR per example is illustrated for the proposed approach and the baselines. Note that the average GIR per example for Lasso and all baselines that use the same number of features is a single number. We observe that the proposed approach achieves similar or much better average GIR per example compared to ETANA. Further, in comparison to Lasso and methods that use all features, the proposed approach achieves larger average GIR per example, as seen by looking at the median and the interquartile range. This experiment validates the importance of instance–wise feature acquisition.

The observations made confirm that the inclusion of powerful classifiers enhances accuracy. Moreover, the proposed algorithms seem to attain a good balance between accuracy and the average number of acquired features by forwarding difficult–to–classify examples to any of the powerful classifiers (see also Fig. 5). This is crucial in real–world applications, where the acquisition of features is either prohibitive (e.g., due to cost or unwillingness of users to provide sensitive information) or the feature space is large.

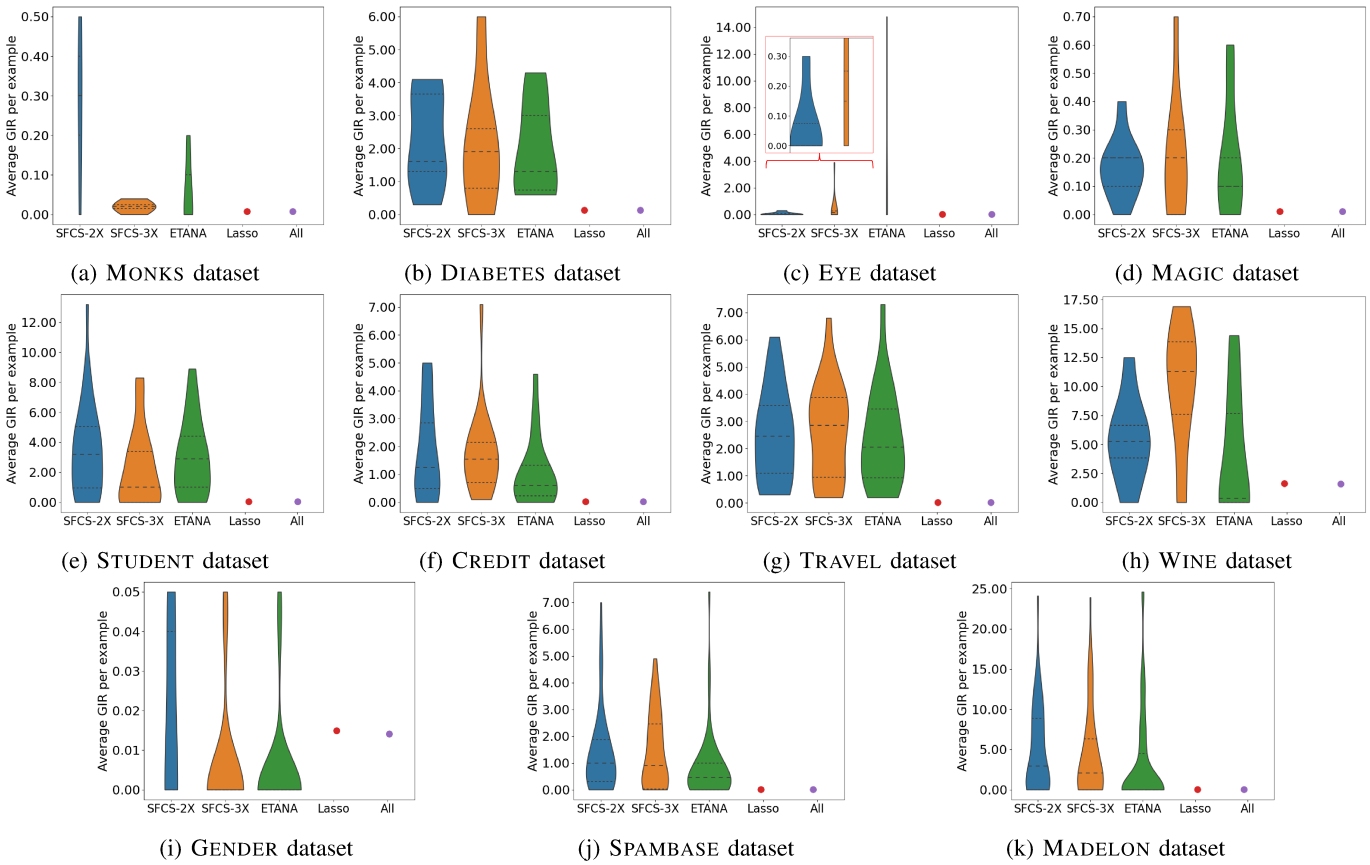
In order to evaluate the statistical significance of the reported outcomes in Table II, we employ the Friedman test, which is commonly used to compare the effectiveness of classifiers across numerous datasets [40]. The average ranks for Acc and Feat are presented in Table II, and the corresponding p–values were found to be 1.29×10^{-3} and 1.12×10^{-19}, respectively. These findings suggest that there exists a notable difference in the performance of SFCS–2X and SFCS–3X, and the baselines considered.

D. Evaluation of SFCS–2X and SFCS–3X

In this section, we analyze the behaviors of SFCS–2X and SFCS–3X during testing. In particular, we start by illustrating and discussing the behavior of SFCS–2X[4] by looking into the evolution of the posterior probability as more features are acquired. We then look into the effect of weighing parameters $\lambda \triangleq \{\lambda_0, \lambda_1, \ldots, \lambda_Z\}$, where $\lambda_0 \triangleq \gamma$, with respect to classifier selection. Finally, we present the distribution of the number of

[4]Similar observations are obtained for SFCS–3X.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2023.3334707



(a) MONKS dataset (b) DIABETES dataset (c) EYE dataset (d) MAGIC dataset

(e) STUDENT dataset (f) CREDIT dataset (g) TRAVEL dataset (h) WINE dataset

(i) GENDER dataset (j) SPAMBASE dataset (k) MADELON dataset

Fig. 4: Distribution of average Gini impurity reduction (GIR) per example. "All" denotes baselines that use all features (e.g., NB, SVM, etc.)

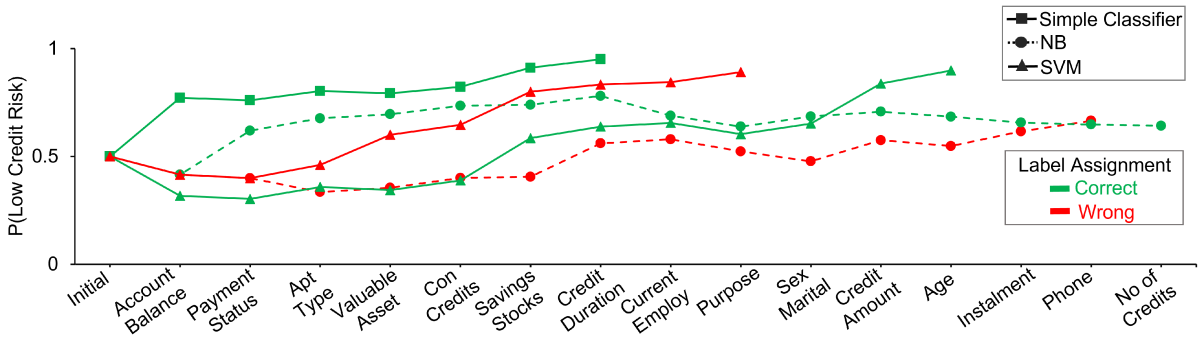


Fig. 5: Posterior probability evolution as more features are acquired for four examples in the CREDIT dataset using SFCS–2X. Green lines indicate correct label assignments, while red lines indicate wrong label assignments.

acquired features during testing for SFCS–2X and SFCS–3X, and discuss our findings.

Fig. 5 demonstrates the evolution of the posterior probability of label assignment as more features are acquired. The reported results are based on five instances of the CREDIT dataset when SFCS–2X is employed. It is observed that in the scenario where the simple classifier is selected, the posterior probability remains significantly away from 0.5 at every step of the feature acquisition process. This suggests that the associated example is relatively easy to classify. Note that when the feature acquisition process terminates, the example is assigned to the class with the highest posterior probability as a result of setting $\Omega_{ij} = 1, \forall i \neq j$ and $\Omega_{ii} = 0, \forall i, j \in \{1, \ldots, N\}$. In contrast, for all other cases, a higher number of features is acquired, while the posterior probability continues to fluctuate around 0.5. This suggests that these examples are much harder–to–classify. Considering the above behavior, SFCS–2X, which sends harder–to–classify examples to one of the powerful classifiers, achieves better performance in comparison to prior work [3], which just employs a simple classifier based on the posterior probability value, as also seen in Table II.

Next, to better understand the classifier selection process, we experiment with different weighing parameter values for

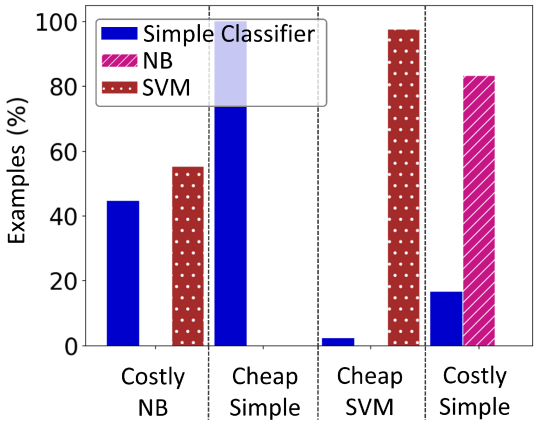


Fig. 6: Effect of $\lambda \triangleq \{\lambda_0, \lambda_1, \lambda_2\}$ in the case of the MAGIC dataset for four cases: (i) Costly NB $\lambda = \{0.288, 0.525, 0.187\}$, (ii) Cheap Simple Classifier $\lambda = \{0.078, 0.312, 0.610\}$, (iii) Cheap SVM $\lambda = \{0.629, 0.281, 0.090\}$, and (iv) Costly Simple Classifier $\lambda = \{0.595, 0.156, 0.249\}$, using SFCS–2X. Note that $\lambda_0 \triangleq 1 - \sum_{z=1}^{2} \lambda_z$ with $\lambda_1, \lambda_2 \in (0, 1)$.

λ and consider four different cases, as shown in Fig. 6 for SFCS–2X. In case (i), the simple and SVM classifiers have relatively comparable weights, while NB has the highest weight, indicating that it is the most expensive to use. In this case, we observe that on average, the simple and SVM classifiers classify 45% and 55% of examples, respectively, while NB is not utilized at all. In case (ii), all examples are classified using the simple classifier, which is relatively cheap to use compared to NB and SVM. Similarly, in case (iii), most examples are classified using SVM, which is the cheapest to use. Finally, in case (iv), the powerful classifiers have lower weights than the simple one but these are still comparable. Thus, we observe that 84% of examples have been classified by the powerful classifiers, specifically NB, which has the least cost, and the rest are forwarded to the simple classifier. Overall, as expected, weighing parameter values influence the classifier selection process during testing, highlighting the need to carefully select such values in practice, especially if any relevant knowledge is available.

Finally, Figs. 7 and 8 provide the distribution of the number of acquired features during testing for the SPAMBASE dataset in the case of SFCS–2X and SFCS–3X, respectively, when $c = 0.00001$, and $\beta = 30$. In Fig. 7, we consider $\lambda = \{0.39, 0.29, 0.32\}$. We see that a simple classifier is selected in most cases where the number of features is lower. On the other hand, when the number of acquired features is higher, those examples are forwarded to NB and SVM. NB is 20% better in accuracy compared to SVM when it is a standalone classifier for SPAMBASE dataset (see Table II). Therefore, NB is preferred compared to SVM for this specific case. In Fig. 8, the simple classifier is selected for fewer features, and examples are forwarded to powerful classifiers when more features are acquired. In SFCS–3X, when comparably equal weights λ_z are assigned to powerful classifiers, we observe that only DT is preferred compared to SVM and NB. At the same time, DT's performance is better when it is a standalone classifier for SPAMBASE dataset (accuracy is 7% and 28% higher than NB and SVM). Therefore, in Fig. 8, even for $\lambda = \{0.59, 0.01, 0001, 0.04\}$, DT is preferred over NB and SVM. We observe that introducing a new powerful classifier does not significantly affect the feature distribution i.e., SFCS–2X and SFCS–3X can classify most examples using the simple classifier with very few features (less than half of the available features). However, when more than half of the features are acquired, SFCS tends to send the corresponding examples to powerful classifiers, indicating that they are harder–to–classify. This insight can tremendously help speed–up the training phase of SFCS by training the powerful classifiers only on this subset of features. Finally, since SFCS acquires a different number of features per example in contrast to the rest of the baselines, its decisions are much easier to interpret.

E. Comparison to Ensemble Learning

Finally, we compare SFCS–2E with often used ensemble learning methods (i) XGB, and (ii) RF (see Table III). We observe that SFCS–2E outperforms both XGB and RF for over half of the datasets (between 1% to 89%) while acquiring 3% to 93% fewer features on average. For the cases where SFCS–2E is outperformed by XGB (1% to 8% drop in accuracy), it acquires 78% to 99% fewer features on average. When outperformed by RF, SFCS–2E acquires 63% to 81% less features. Furthermore, SFCS–2E outperforms both SFCS–2X and SFCS–3X, demonstrating the ability of SFCS to incorporate additional, and more complex classifiers.

Last but not least, Fig. 9 shows that the proposed framework generalizes well for imbalanced datasets (i.e., datasets in which the minority class accounts for $1\% - 39\%$ of the samples), particularly so when using ensemble classifiers (i.e., SFCS–2E). Comparing the accuracy of SFCS–2E (see Table III) with SFCS–2X and SFCS–3X (see Tabel II) further confirms this point.

V. CONCLUSIONS

In this paper, we proposed a method for sequential datum–wise feature acquisition and classification when various classifiers are available. Our proposed approach, SFCS, involves sequentially acquiring features and utilizing them to classify examples using either a simple classifier or one out of a set of powerful classifiers. After validating SFCS on several real–world datasets and comparing it with existing algorithms, we have found that the proposed approach achieves a favorable balance between accuracy and the average number of acquired features. Additionally, the simple classifier is utilized for easy–to–classify examples, while challenging ones are forwarded to the powerful classifiers, thus improving overall accuracy. This observation reinforces the effectiveness of SFCS in practical applications. Furthermore, the selection of classifiers during testing is influenced by the relative weighing parameters.

Nonetheless, the proposed approach has certain limitations that we intend to address in the future. Firstly, there is a training time overhead as we have to train all powerful classifiers for all the possible number of features. For F

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2023.3334707

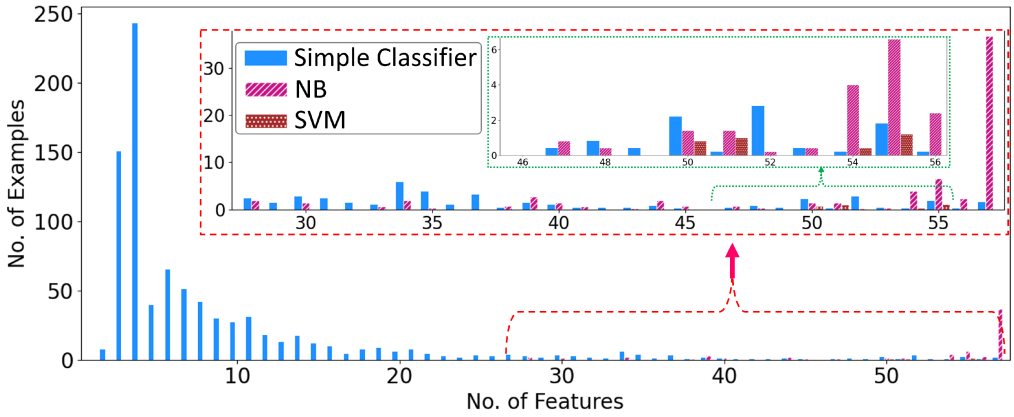


Fig. 7: Distribution of the number of acquired features during testing for the SPAMBASE dataset using SFCS–2X.

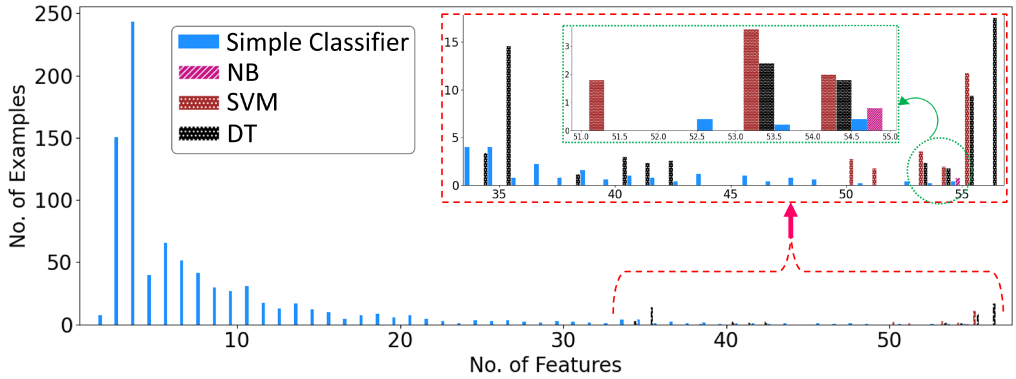


Fig. 8: Distribution of the number of acquired features during testing for the SPAMBASE dataset using SFCS–3X.

TABLE III: Accuracy ("Acc"), average number of acquired features ("Feat"), training ("Train") and testing ("Test") time in seconds for SFCS–2E and ensemble learning methods. Highest and second highest Acc values are indicated as bold and gray–shaded, and gray–shaded, respectively. Smallest Feat values are indicated as bold and gray–shaded.

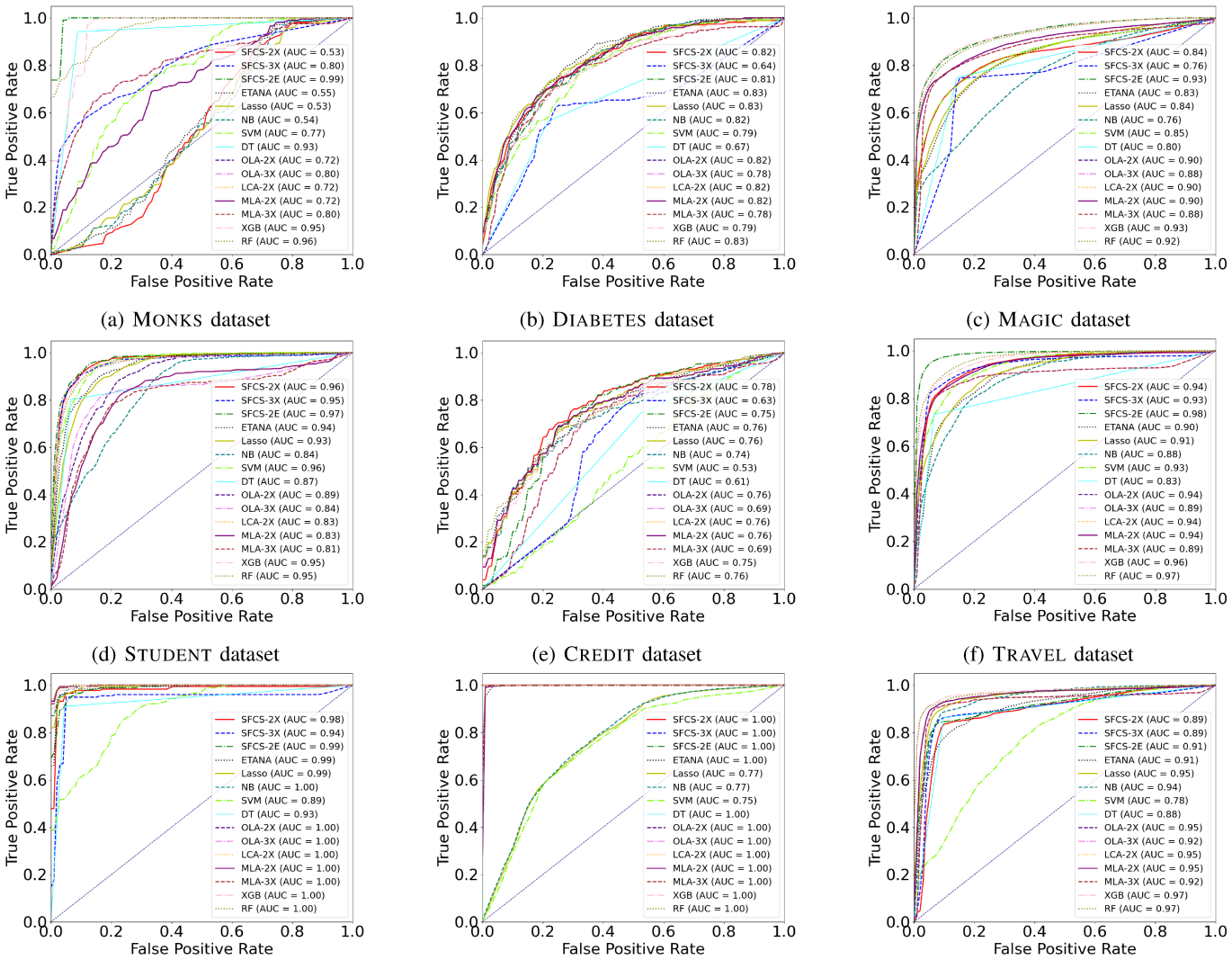
Method	Metric	Monks	Diabetes	Eye	Magic	Student	Credit	Travel	Wine	Gender	Spambase	Madelon
SFCS-2E	Acc	**0.973**	0.744	**0.928**	0.878	**0.858**	**0.768**	**0.896**	0.921	**1.000**	0.896	0.748
	Feat	**6.000**	**2.971**	**12.747**	**9.477**	**2.202**	**15.349**	**15.008**	**2.809**	**1.848**	**10.558**	**7.208**
	Train	2.312	2.931	29.125	62.751	19.043	7.816	91.570	7.273	24.804	62.072	1619.708
	Test	0.259	1.032	9.681	10.977	1.173	0.535	26.177	0.037	0.822	1.239	1.060
XGB	Acc	0.933	0.736	0.491	**0.883**	0.826	0.742	0.801	0.955	**1.000**	**0.932**	**0.815**
	Feat	**6.000**	8.000	14.000	10.000	32.000	20.000	23.000	13.000	20.000	57.000	500.000
	Train	0.094	0.175	1.007	3.235	0.390	0.265	1.818	0.118	0.288	0.902	3.332
	Test	0.002	0.002	0.008	0.010	0.005	0.002	0.004	0.002	0.002	0.005	0.004
RF	Acc	0.875	**0.768**	0.495	0.868	0.831	0.723	0.817	**0.983**	**1.000**	0.925	0.706
	Feat	**6.000**	8.000	14.000	10.000	32.000	20.000	23.000	13.000	20.000	57.000	500.000
	Train	0.321	0.315	1.544	4.639	0.346	0.329	0.938	0.244	0.538	0.814	2.531
	Test	0.017	0.012	0.034	0.039	0.017	0.017	0.017	0.014	0.016	0.022	0.017

features, a classifier $C_z, z = 1, \ldots, Z$, is trained in total F times. To mitigate this issue, based on the insights gained from Figs. 7 and 8, we plan to explore training powerful classifiers using a subset of features. We also plan to investigate the potential of using a multi–armed bandit approach to expedite the training process. Secondly, the accuracy of the proposed approach is affected by the ordering of features. With F features involved, there exist $F!$ possible feature orderings. Thus, we intend to extend this work to enable the acquisition of features in any order, similar to [41]. Thirdly, the choice of classifier for each example has an impact on overall accuracy. To enhance performance and robustness, we plan to extend the proposed framework to combine decisions from multiple classifiers (i.e., ensemble learning [1], [9]), contrary to just using the decision of one classifier.

APPENDIX A

First, consider the error probability $P_z(e, Y = i|x_1, \ldots, x_S)$ of a given classifier C_z. From Bayes' rule, we have that:

$$P_z(e|Y = i, x_1, \ldots, x_S) = \frac{P_z(e, Y = i|x_1, \ldots, x_S)}{P(Y = i|x_1, \ldots, x_S)}, \quad \text{(A.1)}$$



(a) MONKS dataset (b) DIABETES dataset (c) MAGIC dataset

(d) STUDENT dataset (e) CREDIT dataset (f) TRAVEL dataset

(g) WINE dataset (h) GENDER dataset (i) SPAMBASE dataset

Fig. 9: ROC and AUC of the proposed approach and the baselines for imbalanced datasets.

which implies that:

$$P_z(e, Y = i|x_1, \ldots, x_S) = P_z(e|Y = i, x_1, \ldots, x_S) \times P(Y = i|x_1, \ldots, x_S). \quad \text{(A.2)}$$

At this point, we use Eq. (A.2) to rewrite Eq. (2) as follows:

$$h_S^z = \sum_{i=1}^{N} Q_{S,z}^i \phi_S^i, \quad \text{(A.3)}$$

where $Q_{S,z}^i \triangleq P_z(e|Y = i, x_1, \ldots, x_S)$. Defining $\mathbf{Q}_{S,z} \triangleq [Q_{S,z}^1, \ldots, Q_{S,z}^N]^T$, Eq. (A.3) takes the final form of Eq. (4).

APPENDIX B

For $N = 2$ and $Z = 1$, the optimum classifier selection method becomes:

$$U_S^* = \underset{t\in\{0,1\}}{\operatorname{argmin}}[\lambda_t H_S^t(\phi_S)]. \quad \text{(B.1)}$$

Let the posterior probability be $\phi_S = [p, 1-p]^T$, where $\phi_S^1 = p$, when S features have already been acquired. We assume $Q_{S,1}^i = q_1^i, 0 \leq q_1^i \leq 1, \forall i = 1, 2$, where $\mathbf{Q}_{S,1} = [q_1^1, q_1^2]^T$. We consider $q_1^1 = 0.3$, $q_1^2 = 0.2$ and $\Omega_{ij} = 1, \forall i \neq j$, $\Omega_{ii} = 0, \forall i, j \in \{1, 2\}$. Consider $\gamma = \lambda_0 = 1 - \lambda_1, 0 < \lambda_1 < 1$ (see Eq. (1)), and let us simplify the form of the cost curves $\lambda_t H_S^t(\phi_S)$. Specifically, for $t = 0$, we have:

$$\lambda_0 H_S^0(\phi_S) = \begin{cases} \lambda_0 p, & 0 \leq p \leq 0.5 \\ \lambda_0(1-p), & 0.5 < p \leq 1 \end{cases} \quad \text{(B.2)}$$

On the other hand, for $t = 1$, we have:

$$\lambda_1 H_S^1(\phi_S) = \lambda_1(q_1^1 p + q_1^2(1-p)) = \lambda_1(q_1^1 - q_1^2)p + \lambda_1 q_1^2. \quad \text{(B.3)}$$

As shown in Fig. 1, the cost curves in Eqs. (B.2) and (B.3) may intersect, thus creating regions in the posterior probability p space, where the simple classifier is preferred compared to the powerful classifier and vice versa. In particular, when the posterior probability p falls in either of the intervals $[0, \alpha_1]$ and $[\alpha_2, 1]$ (referred to as region A in Fig. 1), the simple classifier is selected (i.e., $U_S^* = 0$). In contrast, when the

posterior probability p falls in the interval (α_1, α_2), where $0 \leq \alpha_1 < 0.5 < \alpha_2 \leq 1$, the powerful classifier is selected (i.e., $U_S^* = 1$). Next, we determine the conditions that $\lambda_1 \in (0, 1)$ needs to satisfy, so that these regions exist.

We start by looking at the interval $[0, 0.5)$ and the intersection of the curves described by Eqs. (B.2) (upper part) and (B.3). Specifically, we determine the point of intersection by solving for a_1 as follows:

$$\begin{aligned}
\lambda_0\alpha_1 &= \lambda_1(q_1^1 - q_1^2)\alpha_1 + \lambda_1 q_1^2 \rightarrow \\
(1-\lambda_1)\alpha_1 &= \lambda_1(q_1^1 - q_1^2)\alpha_1 + \lambda_1 q_1^2 \rightarrow \\
\alpha_1 &= \frac{\lambda_1 q_1^2}{(1-\lambda_1) - \lambda_1(q_1^1 - q_1^2)}. \qquad \text{(B.4)}
\end{aligned}$$

We compute λ_1 such that $\alpha_1 \in [0, 0.5)$. Specifically, $\lambda_1 q_1^2 \geq 0$ because $\lambda_1 \in (0, 1)$ and $q_1^2 \in [0, 1]$. Therefore to satisfy $\alpha_1 \geq 0$, the denominator of Eq. (B.4) should be:

$$\begin{aligned}
(1-\lambda_1) - \lambda_1(q_1^1 - q_1^2) > 0 \rightarrow \\
1 - \lambda_1(1 + q_1^1 - q_1^2) > 0 \rightarrow \\
1 > \lambda_1(1 + q_1^1 - q_1^2) \rightarrow \\
\lambda_1 < \frac{1}{1 + q_1^1 - q_1^2} \triangleq \mu_1. \qquad \text{(B.5)}
\end{aligned}$$

Also to satisfy $\alpha_1 < 0.5$, Eq. (B.4) should be:

$$\begin{aligned}
\frac{\lambda_1 q_1^2}{(1-\lambda_1) - \lambda_1(q_1^1 - q_1^2)} < 0.5 \rightarrow \\
\lambda_1 q_1^2 < 0.5(1 - \lambda_1(1 + q_1^1 - q_1^2)) \rightarrow \\
\lambda_1 q_1^2 + 0.5\lambda_1(1 + q_1^1 - q_1^2) < 0.5 \rightarrow \\
\lambda_1(q_1^2 + 0.5(1 + q_1^1 - q_1^2)) < 0.5 \rightarrow \\
\lambda_1 < \frac{0.5}{q_1^2 + 0.5(1 + q_1^1 - q_1^2)} \rightarrow \\
\lambda_1 < \frac{1}{2q_1^2 + 1 + q_1^1 - q_1^2} \rightarrow \\
\lambda_1 < \frac{1}{1 + q_1^1 + q_1^2} \triangleq \mu_2. \qquad \text{(B.6)}
\end{aligned}$$

Next, we look at the interval $(0.5, 1]$ and the intersection of the curves described by Eqs. (B.2) (lower part) and (B.3). Specifically, we determine the point of intersection by solving for α_2 as follows:

$$\begin{aligned}
\lambda_0(1-\alpha_2) &= \lambda_1(q_1^1 - q_1^2)\alpha_2 + \lambda_1 q_1^2 \rightarrow \\
(1-\lambda_1)(1-\alpha_2) &= \lambda_1(q_1^1 - q_1^2)\alpha_2 + \lambda_1 q_1^2 \rightarrow \\
\alpha_2 &= \frac{(1-\lambda_1) - \lambda_1 q_1^2}{(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2)} \\
&= 1 - \frac{\lambda_1 q_1^1}{(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2)}. \qquad \text{(B.7)}
\end{aligned}$$

We compute λ_1 such that $\alpha_2 \in (0.5, 1]$. Therefore, to satisfy $0.5 < \alpha_2$, Eq. (B.7) should be:

$$\begin{aligned}
0.5 < 1 - \frac{\lambda_1 q_1^1}{(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2)} \rightarrow \\
\frac{\lambda_1 q_1^1}{(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2)} < 0.5 \rightarrow \\
\lambda_1 q_1^1 < 0.5(1 - \lambda_1 + \lambda_1(q_1^1 - q_1^2)) \rightarrow \\
\lambda_1 q_1^1 < 0.5(1 - \lambda_1(1 - q_1^1 + q_1^2)) \rightarrow \\
2\lambda_1 q_1^1 < 1 - \lambda_1(1 - q_1^1 + q_1^2) \rightarrow \\
2\lambda_1 q_1^1 + \lambda_1(1 - q_1^1 + q_1^2) < 1 \rightarrow \\
\lambda_1(1 + q_1^1 + q_1^2) < 1 \rightarrow \\
\lambda_1 < \frac{1}{1 + q_1^1 + q_1^2} \triangleq \mu_3. \qquad \text{(B.8)}
\end{aligned}$$

Also to satisfy $\alpha_2 \leq 1$, Eq. (B.7) should be:

$$\begin{aligned}
1 - \frac{\lambda_1 q_1^1}{(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2)} \leq 1 \rightarrow \\
0 \leq \frac{\lambda_1 q_1^1}{(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2)}. \qquad \text{(B.9)}
\end{aligned}$$

Here, $\lambda_1 \in (0, 1)$ and $q_1^1 \in [0, 1]$. Therefore, $\lambda_1 q_1^1 \geq 0$. To satisfy Eq. (B.9), the following inequality must hold:

$$\begin{aligned}
(1-\lambda_1) + \lambda_1(q_1^1 - q_1^2) > 0 \rightarrow \\
1 - \lambda_1(1 - q_1^1 + q_1^2) > 0 \rightarrow \\
\lambda_1 < \frac{1}{1 - q_1^1 + q_1^2} \triangleq \mu_4. \qquad \text{(B.10)}
\end{aligned}$$

Finally, for α_1 and α_2 to satisfy $0 \leq \alpha_1 < 0.5 < \alpha_2 \leq 1$, λ_1 should satisfy $\lambda_1 < \mu_1, \lambda_1 < \mu_2, \lambda_1 < \mu_3$ and $\lambda_1 < \mu_4$. Therefore using Eqs. (B.5), (B.6), (B.8) and (B.10), we choose λ_1 such that:

$$\begin{aligned}
&\lambda_1 < \min[\mu_1, \mu_2, \mu_3, \mu_4] \rightarrow \\
&\lambda_1 < \min[\mu_1, \mu_2, \mu_4], \mu_2 = \mu_3 \rightarrow \\
&\lambda_1 < \min\left[\frac{1}{1 + q_1^1 - q_1^2}, \frac{1}{1 + q_1^1 + q_1^2}, \frac{1}{1 - q_1^1 + q_1^2}\right] \rightarrow \\
&\lambda_1 < \frac{1}{1 + q_1^1 + q_1^2}. \qquad \text{(B.11)}
\end{aligned}$$

For this example, we select $\lambda_1 = 0.5$ where $\lambda_0 = 1 - \lambda_1$ (see Eq. (1)).

REFERENCES

[1] K. P. Murphy, *Probabilistic machine learning: an introduction*. MIT press, 2022.

[2] X. Hu, P. Zhou, P. Li, J. Wang, and X. Wu, "A survey on online feature selection with streaming features," *Frontiers of Computer Science*, vol. 12, no. 3, pp. 479–493, 2018.

[3] Y. W. Liyanage, D.-S. Zois, and C. Chelmis, "Dynamic instance-wise joint feature selection and classification," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 169–184, 2021.

[4] ——, "Dynamic instance-wise classification in correlated feature spaces," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 537–548, 2021.

[5] M. Moradi, Y. Chen, X. Du, and J. M. Seddon, "Deep ensemble learning for automated non-advanced amd classification using optimized retinal layer segmentation and sd-oct scans," *Computers in Biology and Medicine*, p. 106512, 2023.

[6] C. Xu, C. Fu, W. Liu, S. Sheng, and S. Yang, "Data-driven decision model based on dynamical classifier selection," *Knowledge-Based Systems*, vol. 212, p. 106590, 2021.
[7] A. Maciel-Guerra, G. P. Figueredo, E. Marti, M. J. Alcocer, and J. Twycross, "Subspace-based dynamic selection: a proof of concept using protein microarray data," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.
[8] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.
[9] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.
[10] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195–216, 2018.
[11] J. Elmi, M. Eftekhari, A. Mehrpooya, and M. R. Ravari, "A novel framework based on the multi-label classification for dynamic selection of classifiers," *International Journal of Machine Learning and Cybernetics*, pp. 1–18, 2023.
[12] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *arXiv preprint arXiv:1811.12808*, 2018.
[13] Z.-H. Zhou, "Ensemble learning," in *Machine learning*. Springer, 2021, pp. 181–210.
[14] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.
[15] D. Ruta and B. Gabrys, "An overview of classifier fusion methods," *Computing and Information systems*, vol. 7, no. 1, pp. 1–10, 2000.
[16] R. M. Cruz, D. V. Oliveira, G. D. Cavalcanti, and R. Sabourin, "Fire-des++: Enhanced online pruning of base classifiers for dynamic ensemble selection," *Pattern Recognition*, vol. 85, pp. 149–160, 2019.
[17] J. Elmi and M. Eftekhari, "Multi-layer selector (mls): Dynamic selection based on filtering some competence measures," *Applied Soft Computing*, vol. 104, p. 107257, 2021.
[18] M. Sellmann and T. Shah, "Cost-sensitive hierarchical clustering for dynamic classifier selection," *arXiv preprint arXiv:2012.09608*, 2020.
[19] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 4, pp. 405–410, 1997.
[20] P. C. Smits, "Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 4, pp. 801–813, 2002.
[21] R. M. O. Cruz, L. G. Hafemann, R. Sabourin, and G. D. C. Cavalcanti, "Deslib: A dynamic ensemble selection library in python," *Journal of Machine Learning Research*, vol. 21, no. 8, pp. 1–5, 2020. [Online]. Available: http://jmlr.org/papers/v21/18-144.html
[22] A. Tiwari, "A hybrid feature selection method using an improved binary butterfly optimization algorithm and adaptive β–hill climbing," *IEEE Access*, 2023.
[23] H. Singh and O. Arandjelović, "Data efficient support vector machine training using the minimum description length principle," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 1361–1365.
[24] O. Queen and S. J. Emrich, "Lasso-based feature selection for improved microbial and microbiome classification," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2021, pp. 2301–2308.
[25] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.
[26] N. AlNuaimi, M. M. Masud, M. A. Serhani, and N. Zaki, "Streaming feature selection algorithms for big data: A survey," *Applied Computing and Informatics*, 2019.
[27] J. Zhou, D. Foster, R. Stine, and L. Ungar, "Streaming feature selection using alpha-investing," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 384–393.
[28] S. Perkins, K. Lacker, and J. Theiler, "Grafting: Fast, incremental feature selection by gradient descent in function space," *The Journal of Machine Learning Research*, vol. 3, pp. 1333–1356, 2003.
[29] J. Yoon, J. Jordon, and M. van der Schaar, "INVASE: Instance-wise variable selection using neural networks," in *International Conference on Learning Representations*, 2018.
[30] Q. Xiao and Z. Wang, "Mixture of deep neural networks for instance-wise feature selection," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 917–921.
[31] S. P. Ekanayake, Y. W. Liyanage, and D.-S. Zois, "Dynamic feature selection for classification in structured environments," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2021, pp. 140–144.
[32] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005, vol. 1.
[33] 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, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[34] M. Claesen, F. De Smet, J. A. Suykens, and B. De Moor, "Fast prediction with svm models containing rbf kernels," *arXiv preprint arXiv:1403.0736*, 2014.
[35] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013. [Online]. Available: https://www.openml.org/
[36] "Kaggle," https://www.kaggle.com.
[37] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[38] S. Nembrini, I. R. König, and M. N. Wright, "The revival of the gini importance?" *Bioinformatics*, vol. 34, no. 21, pp. 3711–3718, 2018.
[39] G. K. Rajbahadur, S. Wang, G. A. Oliva, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *IEEE Transactions on Software Engineering*, vol. 48, no. 7, pp. 2245–2261, 2021.
[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[41] Y. W. Liyanage and D.-S. Zois, "Optimum feature ordering for dynamic instance–wise joint feature selection and classification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3370–3374.

Sachini Piyoni Ekanayake received the B.Sc. degree in electrical and electronic engineering from the University of Peradeniya, Sri Lanka, in 2017. Currently, she is working toward the Ph.D. degree in electrical and computer engineering at the University at Albany, State University of New York, USA. Her research interests include machine learning and statistical signal processing.

Daphney-Stavroula Zois received the B.S. degree in computer engineering and informatics from the University of Patras, Patras, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA. Previous appointments include the University of Illinois, Urbana–Champaign, IL, USA. She is an Associate Professor in the Department of Electrical and Computer Engineering, University at Albany, State University of New York, Albany, NY, USA. She received the Viterbi Dean's and Myronis Graduate Fellowships, the NSF CAREER award, and a Google AI for Social Good Impact Scholars award. She has served and is serving as Co-Chair, TPC member or reviewer in international conferences and journals, such as AAAI, ICASSP, ICLR, NeurIPS, IEEE TSP, IEEE TIT, IEEE TAI, and IEEE TNNLS. Her research interests include machine learning and statistical signal processing with a particular focus on decision making under uncertainty.

Charalampos Chelmis is an Associate Professor in Computer Science at the University at Albany, State University of New York, and the director of the Intelligent Big Data Analytics, Applications, and Systems Lab. He has served and is serving as Co–Chair, TPC member or reviewer in international conferences and journals including AAAI, TheWebConf, and WSDM. He received the B.S. degree in computer engineering and informatics from the University of Patras, Greece in 2007, and the M.S. and Ph.D. degrees in computer science from the University of Southern California in 2010 and 2013, respectively.