Canadian AI 2022
30 May–3 June 2022, Toronto

Binary Classification with Minimum Observations

Shamir Khandaker*, Aminul Islam
School of Computing and Informatics
University of Louisiana at Lafayette
Lafayette, LA 70504

Abstract

Binary classification with minimum observations is an important task in applications where enough training data are not available. In this paper, we propose a binary classification approach that is based on an unsupervised ranking algorithm for objects with numerical attributes. The ranking algorithm takes normalized attributes of numerical objects as input and returns the weights of attributes. These weights are used to rank the objects. We propose a class labelling algorithm that labels each side of the ranked objects as a class using less than or equal to 15 labeled data objects or observations. Evaluation on six different data sets shows that the proposed approach is comparable to the state-of-the-art binary classification algorithms that use 70 percent observations compared to less than one percent observations (on average) used by the proposed approach.

Keywords: Binary classification, training, minimum data

1. Introduction

Binary classification with minimum observations is an important issue in many applications that do not have large numbers of training data. Some examples of applications could be in elections, medical trials, economic recessions and sporting outcomes. It can also be used to solve problems in materials science [1], to represent inorganic materials [2], predict fundamental properties [3], create atomic potential [4]. We can furthermore use it to identify fundamental candidates [5], analyze complex reaction networks and guide experimental designs [6].

Insufficient training data causes large accuracy/performance drops in most classification algorithms [7]. Figure 1 shows how the performance (y-axis) of a traditional machine learning classifier is related to training sample sizes (x-axis).

From Figure 1, it is clear that for traditional machine learning classifiers, the relationship between performance and training sample sizes is logarithmic. This means that till certain point or threshold the accuracy/performance grows according to a power law and then reaches a plateau [9]. This threshold widely varies based on domain and data sets. However, in this paper, we define 'minimum' as a number of observation below that threshold (more specifically, less than or equal to 15). For example, [10] showed that binary logistic regression models based on 400 observations are not even dependable. There is a broad

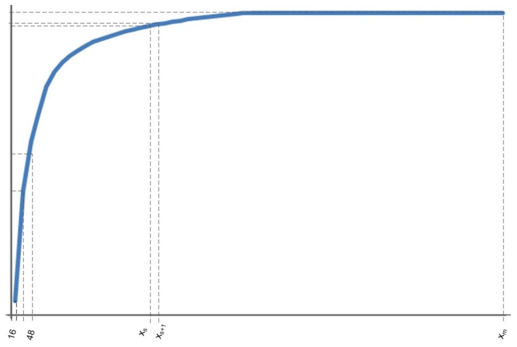


Figure 1. Generic learning curve for classification problem (taken from [8]).

*shamir-towsif.khandaker1@louisiana.edu

agreement among researchers that binary logistic regression models that use less than 200 observations are highly unreliable.

That is why most state-of-the-art binary classification algorithms use 50 to 90% labeled data objects to train the models. These algorithms do not perform well with a small training data. This is because the models produced by these algorithms with a small training data are biased and unstable.

The question is how to find a binary classification method which will use at most 15 observations (i.e., labeled training data) and is comparable to state-of-the-art binary classification methods that require large training samples.

In this paper, we propose a binary classification approach that is based on an unsupervised ranking algorithm for objects with numerical attributes. The ranking algorithm takes normalized attributes of numerical objects as input and returns the weights of attributes. These weights are used to rank the objects. We also propose a class labelling algorithm that labels each end of the ranked objects as one distinct class using less than or equal to 15 labeled data objects or observations.

Overall, we make the following contributions in this work.

(1) We propose a binary classification approach that is based on an unsupervised ranking algorithm. That is, we show how an unsupervised ranking algorithm can be used in binary classification.
(2) We propose a class labeling algorithm that labels top side of the ranked objects as one class and the bottom side as the other class using minimum (less than or equal to 15) observations.
(3) We find an equation that determines the theoretical confidence—in terms of probability—of finding the correct classes using the proposed class labeling algorithm.

The rest of this paper is organized as follows: the related works are discussed in Section 2. Our proposed approach for binary classification with minimum observations is described in Section 3. The experimental results on six datasets and discussion are in Section 4. Direction for future research are described in Section 5, and the paper is concluded.

2. **Related Work**

Alattas et al. [11] propose a new ranking algorithm that is inspired by magnetic properties (MP) and clustering. This algorithm groups the attributes into two clusters (i.e., positive and negative cluster) and place each attribute a weight by using Pearson r correlation coefficient. The idea of using magnetic properties is that if the correlation coefficient between two attributes is positive, it means that they attract each other to be in the same cluster; otherwise, they repulse each other to be in different clusters.

Ranking principal curve (RPC)) [12] model is another ranking algorithm that learns a one-dimensional manifold function using five meta-rules to perform unsupervised ranking tasks on multi-attribute observations. The other work that is closely related to [11] is Li et al. [13], which is based on a two-phase attribute selection procedure. The first phase, Spearman Ranking Correlation Coefficients (SRCC), identifies irrelevant attributes that can adversely affect the ranking, and the second phase uses Extended Fourier Amplitude Sensitivity Test (EFAST) that presents the total effect for each attribute to ranking and then selects the attributes based on those phases. Parameterized function optimization (PFO) [14] evolves an optimization function to learn the weight of the attributes using a Lagrangian function.

Most state-of-the-art methods for ranking numerical objects remove irrelevant attributes and objects with missing values. Alattas et al.'s [11] algorithm can deal with *all attributes*, so there is no need to remove the irrelevant ones.

One recent classification task is fair binary classification. There is a potential risk that sensitive information unfairly influences the outcome of binary classification models. For example if we want to predict whether a university applicant should get scholarship we would like that a model do not use additive sensitive information such as race or gender. Method proposed by [15] uses equal opportunity for binary classification where the method requires true positive rate to be distributed equally across the sensitive groups. We use [15]'s datasets to evaluate our proposed approach.

There are many existing ways to deal with having imbalanced dataset for training such as “Change the loss function”, “Up-sample or Down-sample” [16]. A prominent technique to handle small dataset is to generate synthetic data. In this approach deep learning models are used for generating new synthetic data that are employed in applications such as “image generation”, “video generation” and “image translation” [17]. However, this technique is not useful to generate synthetic numerical data.

3. **Proposed Methodology**

Figure 2 depicts the high-level workflow of our proposed methodology. Given the attribute values of a set of objects, we normalize and compute the correlation coefficient scores for each pair of attributes for all objects. Then we use a modified version of **U**nsupervised **R**anking using **M**agnetic properties and **C**orrelation coefficient (i.e., URMC) algorithm [11] that takes sorted correlation coefficient scores for each pair of attributes from the dataset as an input and returns the weight for each attribute. These attributes' weights are used to rank the objects. We assume that each side of this ranked list represents a class. The Proposed class labelling algorithm uses minimum number of observations and determine the class label. The last algorithm classifies objects based on the class label determined by the previous algorithm.

We use the following notations: Let X refer to a set of n objects, i.e., $X = (x_1, x_2, \ldots, x_i, \ldots, x_n)$ and each of these objects has m number of attributes. Thus, an object x_i can be represented as a set of these attribute values, i.e., $x_i = (a_{i1}, a_{i2}, \ldots, a_{ij}, \ldots, a_{im})$, where a_{ij} refers to the jth attribute value of object x_i. Again, let A_j refer to the set of jth attribute values of all the n objects, i.e., $A_j = (a_{1j}, a_{2j}, \ldots, a_{ij}, \ldots, a_{nj})$.

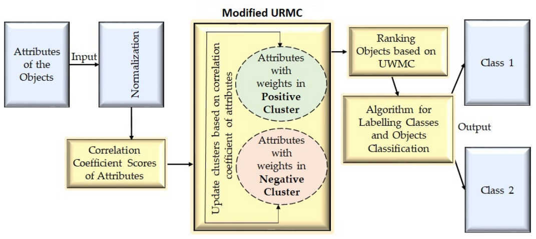

Modified URMC
Attributes of the Objects
Input
Normalization
Correlation Coefficient Scores of Attributes
Update clusters based on correlation coefficient of attributes
Attributes with weights in Positive Cluster
Attributes with weights in Negative Cluster
Ranking Objects based on UWMC
Algorithm for Labelling Classes and Objects Classification
Output
Class 1
Class 2

Figure 2. Steps for binary classification with minimum observations.

3.1. **Normalization**

Normalization is a fundamental step in any ranking algorithm [18, 19]. In general, the range of each attribute values of a dataset widely varies. We use the same approach mentioned in [11] to normalize an attribute value of an object from 1 - 10 using the following Equation 3.1:

$$\text{Normalized Value } (a_{ij}) = 1 + 9 \times \left(\frac{a_{ij} - a_{min}}{a_{max} - a_{min}}\right) \tag{3.1}$$

where a_{ij} is the jth attribute value of object x_i, and a_{max}, a_{min} are the largest and smallest values of jth attribute, respectively.

3.2. **Correlation Coefficient Scores of Attributes**

We already mentioned that A_j refers to the set of jth attribute values of all the n objects (i.e., $A_j = (a_{1j}, a_{2j}, \cdots, a_{ij}, \cdots, a_{nj})$) and each of these objects has m number of attributes. For each A_j against the rest, we compute $\frac{m^2-m}{2}$ attribute-pair correlation coefficient scores (shown as grey cells in Figure 3). That is, r_{ij} is the correlation coefficient between A_i and A_j. Using the special character of symmetric matrices can save time and storage during the attribute-pair correlation coefficient computation.

Generally, a square matrix of order m requires storage for m^2 elements. If it is a symmetric matrix then it can be stored in about less than half the space, $\frac{m^2-m}{2}$ elements (shown as grey cells in Figure 3). Only the upper (or lower) triangular elements of the matrix excluding the main diagonal need to be explicitly stored. The implicit elements of the matrix can be retrieved interchanging row and column numbers. An efficient data structure for storing a symmetric matrix is a simple linear array. If the upper triangular elements of the matrix excluding the main diagonal are retained, the linear array, R, is organized as: $R = \{r_{12}, r_{13}, \cdots, r_{1m}, r_{23}, \cdots, r_{2m}, \cdots, r_{(m-1)m}\}$. We sort R in descending order which is the input to the next URMC algorithm. The indexing rule to retrieve the element r_{ij} from R is [20]:

$$
\begin{array}{c|cccccc}
 & A_1 & A_2 & A_3 & \cdots & A_{m-1} & A_m \\
\hline
A_1 & r_{11} & r_{12} & r_{13} & \cdots & r_{1(m-1)} & r_{1m} \\
A_2 & r_{21} & r_{22} & r_{23} & \cdots & r_{2(m-1)} & r_{2m} \\
A_3 & r_{31} & r_{32} & r_{33} & \cdots & r_{3(m-1)} & r_{3m} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
A_{m-1} & r_{(m-1)1} & r_{(m-1)2} & r_{(m-1)3} & \cdots & r_{(m-1)(m-1)} & r_{(m-1)m} \\
A_m & r_{m1} & r_{m2} & r_{m3} & \cdots & r_{m(m-1)} & r_{mm}
\end{array}
$$

Figure 3. Symmetric matrix of attributes' correlation coefficient.

$$r_{ij} \leftarrow R[(i-1)(m-1) - (i-1)i/2 + j - 1] \qquad (3.2)$$

3.3. **Modified URMC Algorithm**

The original URMC algorithm [11] clusters the attributes into similar groups and updates the weight of attributes that can be used to rank the objects. The original URMC algorithm takes $A_1, A_2, \cdots, A_j, \cdots, A_m$ as input where A_j is the set of jth attribute values of all the n objects (i.e., $A_j = (a_{1j}, a_{2j}, \cdots, a_{ij}, \cdots, a_{nj})$). In our modified version, the sorted R computed in the previous section (i.e., Section 3.2) is used as input. The reason of this modification is that using the attribute-pairs with higher correlation coefficient earlier in the algorithm makes the higher weight attributes stable earlier which has an impact on the final weight of the attributes. In other words, feeding the algorithm the sorted correlation coefficient of attributes rather than in random order has an impact on the final weight of each attribute.

We already mentioned that the modified URMC algorithm takes correlation coefficient scores of attributes of a dataset ordered from high to low and assigns each attribute to a positive or negative cluster with weights. Algorithm 1 demonstrates the pseudo code of the modified URMC algorithm which is based on magnetic properties and the correlation coefficients between all possible pairs of attributes. Initially, all the attributes are set in the positive cluster with weight 0. If the correlation coefficient is positive between two attributes, it signifies that they attract each other to be in the same cluster; otherwise, they repel to be in different clusters.

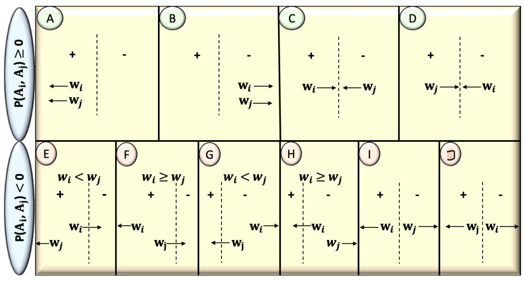


Figure 4. A comprehensive overview of URMC algorithm (taken from [11]).

A comprehensive overview of URMC algorithm is shown in Figure 4 which is split into two parts: top part with a positive r_{ij} (i.e., $r(A_i, A_j) \geq 0$) and bottom part with a negative r_{ij} (i.e., $r(A_i, A_j) < 0$) between attributes, A_i and A_j.

Cells A through D represent the top part with positive r between the attributes (Line 4 - 17, Algorithm 1). When two attributes are in the same cluster (either positive or negative), positive r between the two attributes means that they attract each other to be in the same cluster with more weights. Now, if the r between two attributes is positive and they are in different clusters, it means that they attract each other to bring the other in its own cluster.

Cell A shows that if attributes w_i and w_j are in the positive cluster and their r is positive, then they should be in the positive cluster and their weight will be updated by adding the r to their previous weights. This represents the concept that both attributes attract each other to be more positive if they were in the positive cluster and their r is positive (Line 5 - 7 Algorithm 1). Cell B shows that if attributes w_i and w_j are in the negative cluster and their r is positive, then they should be in the negative cluster and their weight will be updated by subtracting the r from their previous weights. This shows that both attributes attract each other to be more negative if they were in the negative cluster and their r is positive (Line 8 - 10, Algorithm 1).

Cell C shows that if attribute w_i and w_j are in the positive and negative clusters, respectively and their r is positive, then w_i attracts w_j to be in the positive cluster and w_j attracts w_i to be in the negative cluster. Thus, the weight of w_i will be updated by subtracting the r from its previous weight. And the weight of w_j will be updated by adding the r to its previous weight (Line 11 - 13, Algorithm 1). The idea of cell D is similar to that of cell C.

On the other hand, cells E through J represent the bottom part with negative r between the attributes (Line 18 - 42, Algorithm 1). In this part, since the r between two attributes is negative, it means that the two attributes repulse each other to be in different clusters.

Cell E shows that if attributes w_i and w_j are in the positive cluster and the weight of w_i is less than that of w_j (i.e., $w_i < w_j$) and their r is negative, then w_i and w_j repulse each other to be in different clusters. Thus, the weight of w_i will be updated by adding the r to its previous weight. As the r is negative, adding it to the previous weight of w_i will shift w_i towards the negative cluster. And the weight of w_j will be updated by subtracting the r from its previous weight. Again, as the r is negative, subtracting it from the previous weight of w_j will move

Algorithm 1 : Modified URMC Algorithm.

```
INPUT: Sorted $R$
OUTPUT: $W = (w_1, w_2, \dots, w_j, \dots, w_m)$
1: $W \leftarrow 0$        ▷ initialize attributes' weight with 0
2: for $i = 1$ to $m$ do
3:     for $j = i + 1$ to $m$ do
4:         if $r_{ij} >= 0$ then
5:             if ($w_i$>=0 && $w_j$>=0) then
6:                 $w_i \leftarrow w_i + r_{ij}$
7:                 $w_j \leftarrow w_j + r_{ij}$
8:             else if ($w_i$<0 && $w_j$<0) then
9:                 $w_i \leftarrow w_i - r_{ij}$
10:                $w_j \leftarrow w_j - r_{ij}$
11:            else if $w_i$>=0 && $w_j$<0 then
12:                $w_i \leftarrow w_i - r_{ij}$
13:                $w_j \leftarrow w_j + r_{ij}$
14:            else
15:                $w_i \leftarrow w_i + r_{ij}$
16:                $w_j \leftarrow w_j - r_{ij}$
17:            end if
18:        else
19:            if ($w_i$>=0 && $w_j$>=0) then
20:                if $w_i < w_j$ then
21:                    $w_i \leftarrow w_i + r_{ij}$
22:                    $w_j \leftarrow w_j - r_{ij}$
23:                else
24:                    $w_i \leftarrow w_i - r_{ij}$
25:                    $w_j \leftarrow w_j + r_{ij}$
26:                end if
27:            else if $w_i$<0 && $w_j$<0 then
28:                if $w_i < w_j$ then
29:                    $w_i \leftarrow w_i + r_{ij}$
30:                    $w_j \leftarrow w_j - r_{ij}$
31:                else
32:                    $w_i \leftarrow w_i - r_{ij}$
33:                    $w_j \leftarrow w_j + r_{ij}$
34:                end if
35:            else if $w_i$>=0 && $w_j$<0 then
36:                $w_i \leftarrow w_i - r_{ij}$
37:                $w_j \leftarrow w_j + r_{ij}$
38:            else
39:                $w_i \leftarrow w_i + r_{ij}$
40:                $w_j \leftarrow w_j - r_{ij}$
41:            end if
42:        end if
43:    end for
44: end for
```

w_j towards more positive side (Line 19 - 22, Algorithm 1). The idea of cell F is similar to that of cell E.

Cell G shows that if attributes w_i and w_j are in the negative cluster and the weight of w_i is less than that of w_j (i.e., $w_i < w_j$) and their r is negative, then w_i and w_j repulse each other to be in different clusters. Thus, the weight of w_i will be updated by adding the r to its previous weight. And the weight of w_j will be updated by subtracting the r from its previous weight (Line 27 - 30, Algorithm 1). The idea of cell H is similar to that of cell G.

Cell I shows that if attributes w_i and w_j are in the positive and negative clusters, respectively and their r is negative, then w_i and w_j repulse each other to be in different cluster. Thus, the weight of w_i will be updated by subtracting the negative r from its previous weight. And the weight of w_j will be updated by adding the r to its previous weight (Line 35 - 37, Algorithm 1). It means that w_i and w_j will move towards more positive and negative side of the cluster, respectively. The idea of cell J is similar to that of cell I.

3.4. **Ranking Objects based on URMC**

The ranking score (RS) of an object, x_i, is computed using the URMC weights of all the attributes from Algorithm 1 in Section 3.3 by the following equation [11]:

$$\mathrm{RS}(x_i) = w_1 \times a_{i1} + w_2 \times a_{i2} \dots + w_j \times a_{ij} \dots + w_m \times a_{im}$$

where w_j is the URMC weight of attribute j and a_{ij} is the jth attribute value of object x_i. Sorting these scores in descending order with index will provide the ranked list of objects.

3.5. **Algorithm for Labelling Classes**

The idea is that the top ranked elements would be in one class (either positive or negative) and the bottom ranked would be in another class. The labels of minimum number of objects (which is ≤ 15) can determine which sides are positive and negative using Algorithm 2. The separation of one class from the other can be done by estimating the percentage of objects in a class from the data.

Algorithm 2 : Class labelling algorithm

INPUT:
scores ← list of tuples containing ranking score and actual index
n ← number of objects
pr ← % of positive targets # default 0.5
train ← number of training data
OUTPUT:
result # if result is true then it means that the positive labels is at the top of the list, otherwise it is at the bottom

```
$ts, fs \leftarrow 0, 0$
$p \leftarrow n * pr$
$m \leftarrow list()$
for $i = 1$ to $range(train)$ do
    $c\_i \leftarrow random(0, n-1)$
    $value, p\_i \leftarrow scores(j)$
    if $c\_i <= (p-1)$ && $p\_i <= (p-1)$ then
        $result \leftarrow True$
        $ts \leftarrow ts + 1$
    else if $c\_i > (p-1)$ && $p\_i > (p-1)$ then
        $result \leftarrow True$
        $ts \leftarrow ts + 1$
    else
        $result \leftarrow False$
        $fs \leftarrow fs + 1$
    end if
    $m.append((p\_i, result))$
end for
sort $m$ in ascending order by $p\_i$
if $ts > fs$ then
    $result \leftarrow True$
else if $ts = fs$ then
    $test_1 \leftarrow$ first previous index of $m$
    $result_1 \leftarrow$ first result of $m$
    $test_2 \leftarrow$ last previous index of $m$
    $result_2 \leftarrow$ last result of $m$
    $d_1 \leftarrow test_1$
    $d_2 \leftarrow n - test_2$
    if $d_1 < d_2$ then
        $result \leftarrow result_1$
    else
        $result \leftarrow result_2$
    end if
else
    $result \leftarrow False$
end if
```

The purpose of Algorithm 2 is to determine on which side of *scores* does the positive labels lie. If at the end of the execution of algorithm, *result* is *True* then it means that positive labels lie at the top of the *scores*. If *result* is *False*, then it is at the bottom.

Algorithm 2 has four inputs. They are *scores*, *n*, *pr*, and *train*. *scores* is the sorted list of tuples where each tuple contains the ranking score and its original index. The list is already sorted on the basis of its score. The value n is the total number of objects/scores. *pr* is the percentage of positive targets. If the percentage of positive targets is not known then the algorithm assume that it is 0.5. *train* is the number of training observations used.

In Line 1, we assign two variables, named ts and fs, with 0. In these variables, we save results that we get for all the tests that have been run in (Line 7- 16). In (Line 7 - 16), we compute *result* and save it in m with the index of the score. If at the end, ts is greater then fs then *result* is *True* as in the case of (Line 20 - 21), otherwise *False* as in the case of (Line 34 - 36). But what if both ts and fs are equal? In (Line 22 - 33), we compare two of the training data from the top-most and the bottom-most sides of the m. Then we take the result of the training data that is nearest to the end side.

If the *result* variable, we get from Algorithm 2 is *True* then it means that the top p objects have been classified into the positive class. Otherwise, it means that the top $n - p$ objects are in the negative class and the rest of them have been classified as positive class. Here, n and p are the number of objects and the number of positive targets, respectively.

3.6. **Algorithm for Calculating Accuracy**

After getting the result from the Algorithm 2, we compute the accuracy of our classification method. Here we iterate through the ranked objects and based on the result that we get from Algorithm 2, we check whether we have correctly classified the object or not.

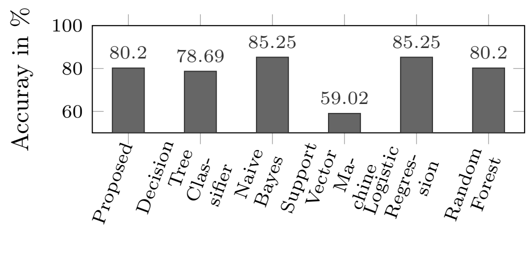
Algorithm 3 has three inputs: *scores* which is the output of Section 3.4, *result* is the output of Algorithm 2 and n is the total number of objects.

In Line 2 - 18, we iterate through the scores list. For each element in the list at first we check the value in the *result*. If it is *True* then it means that the positive labels is at the top. If the current index of each score, which is in the variable i, and the p_i is lower than or equal to $p - 1$ then we increment c_1. If the current index and the p_i is greater than p then we increment c_2. On the other hand, if the *result* is false then we increment c_1 and c_2 for different conditions. We increment c_1 if current index is lower than or equal to $n - p - 1$ and previous index is greater than or equal to $p - 1$. We increment c_2 if current index is greater than $n - p - 1$ and previous index is lower than $p - 1$. At the end we add c_1 and c_2, then multiply it with 100 and divide it by n.

Algorithm 3 : Classification accuracy

INPUT: *result* $\leftarrow$ output from the class labeling algorithm
scores $\leftarrow$ list of tuples containing score, actual index
$n \leftarrow$ number of objects
OUTPUT: *accuracy*

```
1:  $c_1, c_2 \leftarrow 0,\ 0$
2:  for $i$,$value$, $p\_i$ in $enumerate(scores)$ do
3:      if $result$ then
4:          if $i$<=($p$−1) && $p\_i$<=($p$−1) then
5:              $c_1 \leftarrow c_1 + 1$
6:          end if
7:          if $i$>($p$−1) && $p\_i$>($p$−1) then
8:              $c_2 \leftarrow c_2 + 1$
9:          end if
10:     else
11:         if $i$<=($n$−$p$−1) && $p\_i$>=$p$ then
12:             $c_1 \leftarrow c_1 + 1$
13:         end if
14:         if $i > (n-p-1)$ && $p\_i < p$ then
15:             $c_2 \leftarrow c_2 + 1$
16:         end if
17:     end if
18: end for
19: $result \leftarrow (c1 + c2)/n * 100$
```



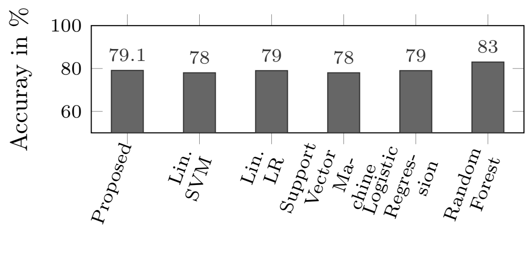


Figure 5. Comparison with different supervised learning models on (a) Heart Disease Dataset, (b) Arrhythmia Dataset

3.7. **Confidence of Finding Correct classes**

Here we find the confidence in terms of probability of finding the correct classes. Suppose the accuracy of binary classification of the proposed method is x. The confidence, C, (i.e., probability) that taking n sample labels (where n is an odd number) can correctly label the classes with accuracy x is:

$$C = \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} x^{n-i}(1-x)^i \tag{3.3}$$

For example, if we assume that the percentage of accuracy of the proposed binary classification method is 80 (i.e., x=0.8), then the confidence that taking 11 sample labels can correctly label the classes is:

$$\begin{aligned} C &= \sum_{i=0}^{5} \binom{11}{i} 0.8^{11-i}(1-0.8)^i \\ &= 0.988 \end{aligned}$$

This could be interpreted as: If we use 11 labels from the dataset and the accuracy of the proposed classification method is 80% then we are 98.8% confident that the label of the proposed class labeling algorithm is correct. In other words, using 11 labels of data object, the probability that our proposed class labeling algorithm can correctly label the classes is 98.8% if the accuracy of the proposed classification method is 80%.

4. **Evaluation and Experimental Results**

For our experiments we have used six datasets. They are as follows: Heart Disease, Arrhythmia, COMPAS, Adult, German, and Drug. The last five datasets are used by [15] for "Fair Binary Classification" task. We have collected these datasets from Kaggle.

4.1. **Heart Disease**

To test our proposed approach, we use Cleveland dataset [21] that contains 14 attributes of 303 candidate heart disease patients. The 14th field (i.e., the "goal" field) refers to the presence (value 1) or absence (value 0) of heart disease in the patient. We use the first 13 attributes as input in our method and the label of the top ranked object (which is 0) to mark that the top ranked patients refers to negative cases. Figure 5(a) shows the accuracy of our method compared to different machine learning measures [22]. Our method outperformed SVM and Decision Tree classifier, and equalled Random Forest.

4.2. **Arrhythmia**

This dataset has 109446 samples and 187 features. Each feature represents a heart beat signal. The target column has values from 0 to 4. Here 0 means 'N', 1 means 'S', 2 means 'V', 3 means 'F' and 4 means 'Q'. When we pre-process the data, we replace all the values

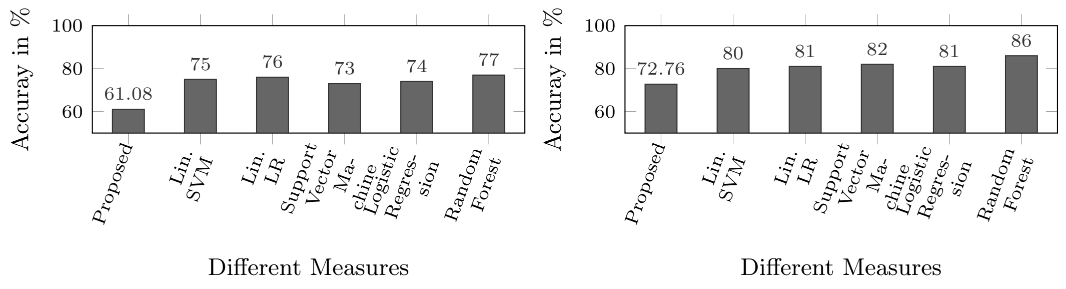


Figure 6. Comparison with different supervised learning models on (a) Compas dataset, (b) Adult dataset

in target which is greater than 0 with 1. After pre-processing we have 18857 samples whose target is positive. The rest are negative. Figure 5(b) shows the accuracy of our method compared to different machine learning measures [22]. Our method outperformed Linear SVM, Linear Logistic Regression, Support Vector Machine and Logistic Regression. It has only been outperformed by Random Forest.

4.3. **COMPAS**

COMPAS dataset has 6172 samples and 11 features. This dataset is used for finding out a person's likelihood of re offending. Features include things such as Two_yr_Recidivism, Number_of_priors etc. It has 2202 samples whose target is positive, Rest are negative. Figure 6(a) shows the accuracy of our method compared to different machine learning measures [22].

4.4. **Adult**

Adult dataset has 48841 samples and 14 features. This dataset is used for finding out a person's likelihood of earning more than 50K. We pre-process this dataset by changing workclass, education, marital-status, occupation, relationship, race, gender, native-country into one-hot vectors [23]. We also use age, fnlwgt, capital-gain, capital-loss, hours-per-week features as it is. It turns our feature size into 107. Of the sample, 76% are less than or equal to 50K. The rest of the 24% are greater than 50K. Figure 6(b) shows the accuracy of our method compared to different machine learning measures [22].

4.5. **German**

German dataset has 1000 samples and 10 features. This dataset is used for finding out whether a person has good or bad credit risk. We pre-process this dataset by changing sex, job, housing, saving accounts, checking account, purpose into one-hot vectors [23]. We also use age, credit amount, duration features as it is. It turns our feature size into 27. Of the sample, 699 are good credit risk. The rest are greater than bad. Figure 7(a) shows the accuracy of our method compared to different machine learning measures [22].

4.6. **Drug**

Drug dataset has 215,063 samples and 2 features. This dataset is used to find out whether the rating of a drug is greater than 5 or not. We pre-process this dataset by changing rating less than or equal to 5 to 0 and the rest to 1. We also convert the date feature to Unix timestamp. 150,768 samples are positive samples. Figure 7(b) shows the accuracy of our method compared to different machine learning measures [22].

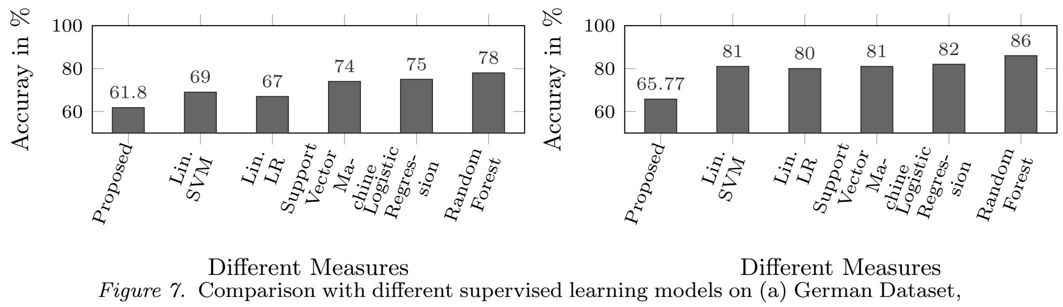


Figure 7. Comparison with different supervised learning models on (a) German Dataset, (b) Drug Dataset

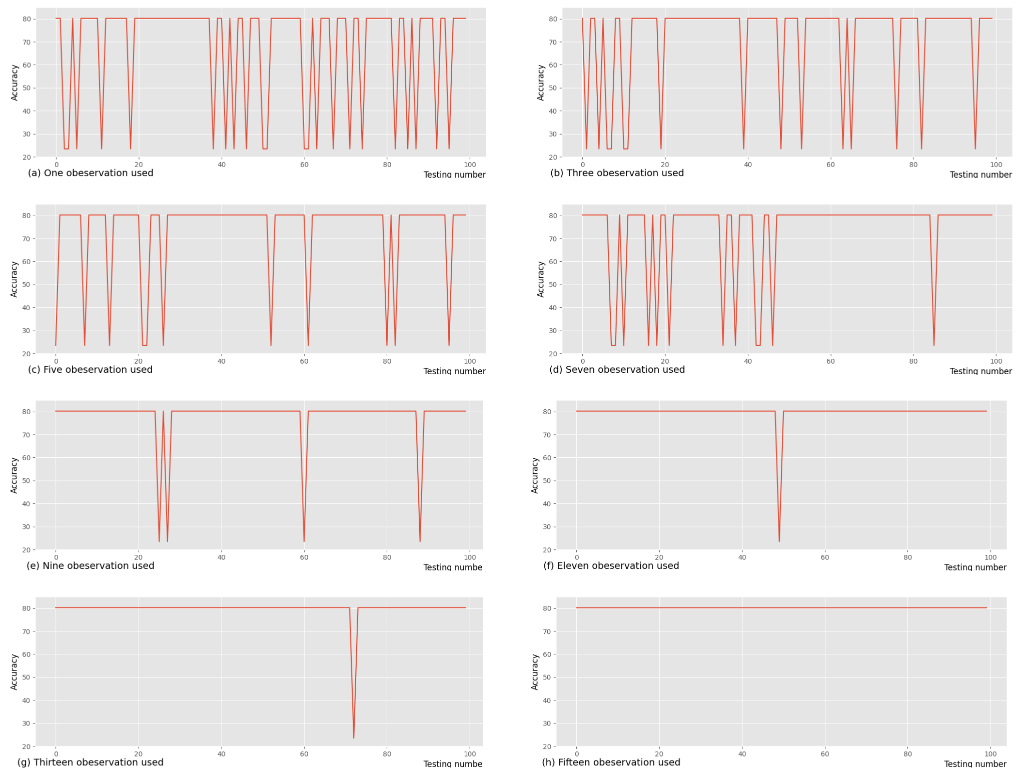


Figure 8. 100 test run of Class Labeling Algorithm on the Heart Disease Dataset using different number of training data.

4.7. **Evaluation of Confidence of Finding Correct classes**

Figure 8 shows the number of times our class labelling algorithm correctly labels the classes for different number of training samples. Each trough in the figure means one instance of incorrect labeling. For example, Figure 8(f) shows that if we use 11 training samples of “Heart Disease” dataset, only one time out of 100 times, our class labelling algorithm incorrectly labels the classes. We run our class labelling algorithm with different number of training samples. We conduct eight experiments shown in Figure 8(a) to (h) where we use 1, 3, 5, 7, 9, 11, 13 and 15 training samples, respectively. As we see from Table 1, the confidence (or the probability) that the class labeling algorithm correctly predicts the classes increases with the increase of the observation/training data. The confidence reaches 100% with 15 training samples. This means that we are 100% confident that if the class labeling algorithm uses 15 training samples, it will correctly label the classes for those data sets shown in Table 1. The practical/experimental results that we get is almost same to the theoretical expected results using Equation 3.3 as shown in Table 1.

4.8. Discussion

The authors of [15] have used 76612, 4320, 34188, 700 and 150544 training samples for "Arrhythmia", "COMPAS", "Adult", "German" and "Drug" data sets, respectively. On the other hand, in our experiments we have used at most 15 training samples as shown in Table 2. For several data sets, the accuracy of our proposed approach is comparable to that of [15]. One of the reasons that we do not get comparable result for the drug dataset is that several features of this data set such as 'review' and 'condition' are text features and we could not incorporate these text features in our algorithm because the ranking algorithm uses only numerical features.

No. of train	Heart Disease		Arrhythmia	
samples	Expected	Practical	Expected	Practical
1	0.80	0.78	0.79	0.78
3	0.89	0.85	0.88	0.89
5	0.94	0.89	0.93	0.97
7	0.97	0.88	0.96	1.00
9	0.98	0.96	0.98	0.99
11	0.99	0.99	0.99	0.99
13	0.99	0.99	0.99	1.00
15	1.00	1.00	1.00	1.00

Table 1. Comparison of the expected confidence values and the experimental/practical confidence values that the class labelling algorithm correctly labels the classes for Heart Disease and Arrhythmia Datasets.

5. Conclusion

This paper shows that an unsupervised ranking algorithm for objects with numerical attributes can be used for binary classification task. One of the strong points of the proposed approach is that it uses a minimum number of observation. For example, in one of the evaluation data set (i.e., Arrhythmia), our proposed approach achieves 79.1 percent accuracy using only 15 observations (i.e., 0.013% of the data set) compared to 83 percent accuracy achieved by a state-of-the-art algorithm that uses 76,612 observations (i.e., 70% of the data set).

Dataset	No. of training samples used by other methods	% of training samples used by other methods	No. of training samples used by our method	% of training samples used by our method
Heart	212	70%	15	4.95%
Arrhythmia	76612	70%	15	0.013%
COMPAS	4320	70%	15	0.243%
Adult	34188	70%	15	0.030%
German	700	70%	15	1.5%
Drug	150544	70%	15	0.006%

Table 2. Comparison of training observations used for different data sets by other methods versus our method.

One of the important future works on this task could be how to use this approach for multi-class classification problem. Another important future work could be how to incorporate text features in the proposed classification approach.

References

[1] Y. Zhang **and** C. Ling. "A strategy to apply machine learning to small datasets in materials science". **in***npj Computational Materials*: 4.1 (2018), **page** 25. ISSN: 2057-3960.

[2] A. Seko, H. Hayashi, K. Nakayama, A. Takahashi **and** I. Tanaka. "Representation of compounds for machine-learning prediction of physical properties". **in***Physical Review B*: 95.14 (2017), **page** 144110.

[3] B. Medasani, A. Gamst, H. Ding, W. Chen, K. A. Persson, M. Asta, A. Canning **and** M. Haranczyk. "Predicting defect behavior in B2 intermetallics by merging ab initio modeling and machine learning". **in***npj Computational Materials*: 2.1 (2016), **pages** 1–10.

[4] C. Chen, Z. Deng, R. Tran, H. Tang, I.-H. Chu **and** S. P. Ong. "Accurate force field for molybdenum by machine learning large materials data". **in***Physical Review Materials*: 1.4 (2017), **page** 043603.

[5] Z. W. Ulissi, A. J. Medford, T. Bligaard **and** J. K. Nørskov. "To address surface reaction network complexity using scaling relations machine learning and DFT calculations". **in***Nature communications*: 8.1 (2017), **pages** 1–7.

[6] P. Raccuglia, K. C. Elbert, P. D. Adler, C. Falk, M. B. Wenny, A. Mollo, M. Zeller, S. A. Friedler, J. Schrier **and** A. J. Norquist. "Machine-learning-assisted materials discovery using failed experiments". **in***Nature*: 533.7601 (2016), **pages** 73–76.

[7] C. Li, J. Wang, L. Wang, L. Hu **and** P. Gong. "Comparison of Classification Algorithms and Training Sample Sizes in Urban Land Classification with Landsat Thematic Mapper Imagery". **in***Remote Sensing*: 6.2 (2014).

[8] R. Figueroa, Q. Zeng-Treitler **and** S. Kandula. "Predicting sample size required for classification performance". **in***BMC Med Inform Decis Mak*: 12.8 (2012).

[9] X. Zhu, C. Vondrick **and** C. Fowlkes. "Do We Need More Training Data?" **in***Int J Comput Vis*: 119 (2016), **pages** 76–92. DOI: https://doi.org/10.1007/s11263-015-0812-2.

[10] M. van Smeden, K. Moons **and** J. de Groot JA. "Sample size for binary logistic prediction models: Beyond events per variable criteria". **in***Stat Methods Med Res*: 28.8 (2019), **pages** 2455–2474. DOI: 10.1177/0962280218784726.

[11] K. Alattas, A. Islam, A. Kumar **and** M. A. Bayoumi. "Unsupervised Ranking of Numerical Observations based on Magnetic Properties and Correlation Coefficient". **in***52nd HICSS Conference*: **byeditor**T. Bui. ScholarSpace, 2019, **pages** 1–10.

[12] C.-G. Li, X. Mei **and** B.-G. Hu. "Unsupervised ranking of multi-attribute objects based on principal curves". **in***Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*: IEEE. 2016, **pages** 1526–1527.

[13] C.-G. Li, X. Mei **and** B.-G. Hu. "Two-Phase Attribute Ordering for Unsupervised Ranking of Multi-Attribute Objects". **in***Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*: IEEE. 2014, **pages** 175–182.

[14] A. Tavanaei, R. Gottumukkalay, A. S. Maida **and** V. V. Raghavan. "Unsupervised Learning to Rank Aggregation using Parameterized Function Optimization". **in***2018 International Joint Conference on Neural Networks (IJCNN)*: IEEE. 2018, **pages** 1–8.

[15] E. Chzhen, C. Denis, M. Hebiri, L. Oneto **and** M. Pontil. "Leveraging Labeled and Unlabeled Data for Consistent Fair Binary Classification". **in***NeurIPS 2019 - 33th Annual Conference on Neural Information Processing Systems*: Vancouver, Canada, **december** 2019.

[16] J. P. Maheswari. *Breaking the curse of small datasets in Machine Learning: Part 1*. https://towardsdatascience.com/breaking-the-curse-of-small-datasets-in-machine-learning-part-1-36f28b0c044d (accessed Jul 1, 2020). San Francisco Bay Area.

[17] A. Torfi **and** E. A. Fox. "CorGAN: Correlation-Capturing Convolutional Generative Adversarial Networks for Generating Synthetic Healthcare Records". **in***The Thirty-Third International FLAIRS Conference (FLAIRS-33)*: 2020, **pages** 335–340.

[18] R. Ginevičius. "Normalization of quantities of various dimensions". **in***Journal of business economics and management*: 9.1 (2008), **pages** 79–86.

[19] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai **and** H. Li. "Learning to rank: from pairwise approach to listwise approach". **in***Proceedings of the 24th international conference on Machine learning*: ACM. 2007, **pages** 129–136.

[20] A. Islam, E. Milios **and** V. Kešelj. "Do Important Words in Bag-of-Words Model of Text Relatedness Help?" **in***Text, Speech, and Dialogue*: **byeditor**P. Král **and** V. Matoušek. Cham: Springer International Publishing, 2015, **pages** 569–577. ISBN: 978-3-319-24033-6.

[21] Cleveland. *Heart Disease Dataset*. https://www.kaggle.com/ronitf/heart-disease-uci (accessed April 20, 2020). University of California Irvine.

[22] ronit. *Heart Disease UCI*. https://www.kaggle.com/vbmokin/heart-disease-comparison-of-20-models (accessed Jul 1, 2020). Bar Ilan University.

[23] C. Guo **and** F. Berkhahn. "Entity embeddings of categorical variables". **in***arXiv preprint arXiv:1604.06737*: (2016).