Discover Computing

Research

Enhanced nearest centroid model tree classifier

Mehmet Hamdi Özçelik[1,2] · **Ekrem Duman**[3] · **Selami Bağrıyanık**[4,5] · **Serol Bulkan**[2]

Received: 26 November 2024 / Accepted: 16 April 2025
Published online: 05 May 2025
© The Author(s) 2025 OPEN

Abstract
In this study, first, we improved an existing variant of the Nearest Centroid algorithm. In this new version, the predictive power of features and within-class variances are used as weights in distance calculation. This version is called the Enhanced Nearest Centroid (ENC). Second, we proposed a new model tree algorithm for binary classification. It is named as the Enhanced Nearest Centroid Model Tree (ENCMT). The model tree is built using ENC at each leaf node of the decision tree. To evaluate the performance of the new model tree, we used an independent test platform and ran the algorithm on 30 binary datasets available therein. Results showed that ENCMT improves the performance of the decision tree algorithm. We also compared ENCMT with the Logistic Model Tree (LMT) algorithm and showed that it outperforms LMT as well. We also designed a bagging algorithm where ENCMT is used to build a random forest. Our comparison results show that its performance is significantly better than the Random Forest (RF) algorithm.

Highlights

- A new model tree algorithm based on the nearest centroid variant
- Very simple but powerful classifier
- Comparisons made at an independent platform over 30 datasets

Keywords Binary classification · Nearest centroid classifier · Model tree · Information value

1 Introduction

One of the main tasks of data mining is predictive modeling, which is related to predicting the future values an attribute (feature, variable) can take based on the information that we have (the values of the other variables). If the variable to be predicted (typically named as the target variable) takes continuous values, regression methods are used, whereas if it takes several discrete values, classification methods are used [1]. If the target variable takes only two different values, the problem is named the binary classification. It is possible to see a lot of use cases of binary

✉ Mehmet Hamdi Özçelik, hamdi.ozcelik@appliedanalytics.com.tr; hamdiozcelik@marun.edu.tr; Ekrem Duman, ekrem.duman@ozyegin.edu.tr; Selami Bağrıyanık, selamibagriyanik@gmail.com; Serol Bulkan, sbulkan@marmara.edu.tr | [1]Applied Analytics AA, Istanbul, Turkey. [2]Department of Industrial Engineering, Marmara Üniversitesi, Mühendislik Fakültesi M5 Blok, Endüstri Mühendisliği Bölümü, Aydınevler Mah, Maltepe, 34840 Istanbul, Turkey. [3]Özyeğin University, Istanbul, Turkey. [4]Bahçeşehir UniversityWingie Enuygun Group, R&D Center, Technology, Istanbul, Turkey. [5]Department of Computer Engineering, İstanbul Health and Technology University, İstanbul, Türkiye.

classification in real life [2]. In machine learning terminology, the classification and regression problems are named supervised learning as we have a target variable with known values (labels) [3].

Binary classification is an old problem for the researchers' community and, surprisingly, the neural network (NN) algorithm. However, it did not gain popularity until the 1990s and is one of the oldest algorithms that was proposed in the 1950s. The logistic regression (logit) algorithm, which takes its roots from the statistics theory, is one of the oldest and most popular classification algorithms [4].

Research on binary classification flourished in the 1980s with the proposal of well-performing decision tree algorithms. One of these algorithms is the CRT (classification and regression trees) algorithm, which can be used for classification and regression purposes [5]. The C4.5 (or C5.0, which is the commercial version of C4.5) and the CHAID (Chi-squared automated interaction detection) algorithms can be listed as the other popular and most successful decision tree algorithms proposed in the 80 s or early 90 s [6, 7]. A variant of decision tree algorithms is the model tree algorithm proposed by Quinlan in 1992 [8]. Originally, they were proposed for regression problems where at each leaf node of the tree, multivariate regression models are built. After a silence, MTs have gained popularity again, and they have been successfully adopted for the classification problems also (e.g. developing a logit model at each leaf node of the decision tree).

The support vector machines (SVMs) introduced in the 1990s took place among the most competitive classification algorithms. Although it was possible to obtain more accurate results with SVMs or NNs, in practice, because of the longer training times or model explainability reasons, people preferred to use decision trees or logit for a long time. However, with the invention of deep neural networks (DNN) in the 2000s, the performance gap from the simple classification algorithms increased, and no matter the explainability issues, DNNs gained high popularity in especially some particular classification problems such as image classification [9].

DT algorithms seemed to be forgotten until the researchers re-invented the importance of bagging and boosting algorithms and applied them to decision trees. Random forests, which consist of many decision trees where each tree is developed using a subset of data and the available features, are reported to outperform simple decision trees [10]. Then, a while later, the so-called XGBoost [11] and LightGBM [12] algorithms used a series of weak learners (very simple decision trees), and with their collective power, they were able to make quite accurate predictions. It has been reported that, in many of the recent KAGGLE competitions, the best results are obtained with such algorithms [13]. So, we can say that fashion is again in favor of simple algorithms, and this was one of our main motivations for searching for a simple but good-performing algorithm in this study.

On the other hand, it has been reported that the kNN algorithm (assigning the class label of a test instance based on the majority of the labels of the *k* nearest instances whose labels are known) is among the best-performing classification algorithms. However, it has a major drawback in that we need to keep all the labeled data in the memory [14].

The nearest centroid (NC) algorithm [15], where first the centroids of the two classes are calculated and then, test instances are assigned the class label whose centroid is closer, is one other simple classifier mentioned in the literature. Motivated by the re-rise of the simple algorithms, we selected the simplest similarity classifier NC as our base classifier.

Similarity classifiers use features without weighting. However, features have different predictive powers and using them as weights at distance calculations improves the performance [16]. In this study, we propose an enhanced version of the NC algorithm based on the logic that the predictive powers of variables are not the same, and thus, they should not have the same amount of say in the distance-to-centroid calculations. We name this enhanced version of the NC algorithm, which can also be regarded as a 1NN variant, as the ENC (Enhanced NC) algorithm.

In real life classification problems, we have really different inherent groupings within the population. In fraud detection problem, various customer profiles exist for legitimate transactions. Similarly, the fraudulent transactions are not all same since they have been initiated from different fraudsters using different methods and they deviate from the victim's behavioral profile in different ways. In customer churn problem, churned customers might leave the company due to many different reasons such as having a pricing issue, an unresolved complaint, a competitors offer etc. Similarly, the customers survived may have many different reasons to keep the relationship with the company. Therefore, using a constant set of weights for features is not enough. Instead, we could divide data into subsets and recalculate weights for each subset. Creating subsets of data could be achieved in a supervised manner using decision trees and this leaded us to the concept of model tree. As a result, we developed Enhanced Nearest Centroid Model Tree (ENCMT) where ENC is used in each leaf of the model tree. We also developed the bagging variant Enhanced Nearest Centroid Model Tree Forest (ENCMTF) where a set of ENCMTs is built to produce aggregate predictions.

To check to performance of these algorithms, we applied them to 30 different binary classification datasets available on an independent test platform [17]. For the comparison metric, we decided to use AUC (area under the ROC curve) since AUC provides an overall measure of performance across all thresholds [18].

The outline of the rest of this study is as follows. In Sect. 2, we review the literature that is closely related to our study. In Sect. 3, we describe the ENC, ENCMT and ENCMTF algorithms together with the necessary background information. The experimental design is explained in Sect. 4, and a discussion of the results obtained is made in Sect. 5. The summary and limitations of the study, together with the directions for possible future work are given in Sect. 6.

2 Related literature

The Nearest Centroid (NC) or Nearest Class Mean (NCM) classification algorithm is one of the simplest classification techniques. It first calculates the centroids for each class and then calculates the distance between the test instance and each of the centroids. It combines key principles of two foundational algorithms: using centroids similar to K-Means and making distance calculations similar to kNN [19]. Hastie et al. [15] show that the performance of the nearest centroid classifier is similar to the one-nearest neighbor and support vector classifier for the protein dataset. They used AUC as the performance metric since it is an overall measure of accuracy. The Nearest Centroid algorithm is simple and used in many areas, along with other algorithms such as intrusion detection [20] and network traffic classification [21]. It is also used with new technologies, such as quantum computing [22].

Nearest shrunken centroid (NSC) classifier [23] is an important variant of Nearest Centroid algorithm which first shrinks each of the class centroids toward the overall centroid for all classes. A robust variant of NSC is developed for heavy-tailed data using Huber loss function [24]. Sahtout et al. [25] extend NSC with hard and order thresholding, and a deep search algorithm to find thresholding value which minimizes cross-validation error over gene expression data sets for human cancers. Neither NC nor NSC take into account the correlation structure among features. Ren et al. [26] brought a solution to this problem by a decorrelation process on tensor data.

The computation time for NC linearly increases with number of features, therefore it is easily used with high-dimensional data. Fraiman and Li [27] developed a new variant of NC for high-dimensional data where they redefine the centroids using disjoint subsets of features and making distance calculations with dimensionality-normalized norm.

Similar to deep neural networks, deep nearest centroid algorithms are emerged. Wang et al. [28] devised deep nearest centroids (DNC) for visual recognition. DNC makes class-wise clustering to discover class sub-centroids. They show that DNC as a nonparametric and explainable algorithm is better than other deep learning methods. Xie et al. [29] proposed a deep centroid variant of NC and used on biomedical omics data. They combined NC with deep cascade strategy in an ensembled approach with random feature scanning.

Modifying the distance calculations against variance is also found in literature [16, 30]. Elen and Avuçlu [30] introduced a variance-sensitive variant of NC which standardizes the distance calculation by standard deviation and z-score and it has better classification accuracy.

Information value and its ingredient Weight of evidence measures are widely used in building scorecards in healthcare [31] and in finance [4]. Feature weighing is an important concept in machine learning [32], and using weights at distance calculations is common in the kNN family of algorithms [33]. Özçelik and Bulkan [16] introduced a new NC algorithm, namely "Nearest Centroid Classifier Based on Information Value and Homogeneity" (NCIVH), using predictive power and within-class homogeneity as weights in the distance calculation. Information Value (IV) is used as the relevance metric, and various metrics are calculated for in-class variance. NCIVH Classifier is also applicable to high-dimensional problems since features that are not important get small weights at similarity computation. Similar to Nearest Shrunken Centroids (NSC), NCIVH considers within-class variance but instead of shrinking the centroids, it uses variance information at similarity calculation. NCIVH does not make any data preprocessing and selects the variance metric based on accuracy. We developed **Enhanced Nearest Centroid (ENC) Classifier** as an improved version of the NCIVH algorithm. The details are given in Sect. 3.

Model trees are ensemble structures where a decision tree with independent basic models at each terminal node (leaf) is built [10]. This idea is initiated by the M5 system, which is proposed by Quinlan [8]. M5 is a decision tree having multivariate linear regression models at each leaf, analogous to piecewise linear functions. Since M5 is suitable only for regression problems, new hybrid algorithms emerged for classification tasks [34]. Kohavi used Naïve Bayes as the base classifier in each leaf to build a model tree, namely NBTree [35] and Logistic Model Trees (LMT) built using logistic regression models at each leaf [36]. In contrast to standard decision tree classifier in which all elements of

each leaf are assigned to a certain class label, i.e., having fixed prediction scores in each leaf, a model tree classifier has several advantages: the ability to use multiple attributes within each leaf, having varying scores in leaves, combining the strength of two algorithms, and a good potential for processing big data. The second algorithm could be integrated into the split decisions (pre-processing) or used after the tree growth (post-processing) [37]. The pruning of the model tree also could be done in a pre-processing or post-processing manner and could lead to more accurate and efficient model trees [38, 39]. There are many applications of model trees to specific domains such as diagnosis of heart failure [40] and steel plates faults prediction [41]. A multi class and multi label version of logistic model trees is also developed [42]. This new method is interpretable, understandable, and explainable. Its performance is measured on 8 datasets and Wilcoxon test statistics is used for significance.

Similar to NBTree and LMT, we developed **an Enhanced Nearest Centroid Model Tree (ENCMT) classifier** where the ENC classifier is used at each leaf. ENCMT first grows a traditional tree and then adds ENC models to each leaf in post-processing.

Model trees could be integrated with Ensemble learning approaches. For example, Moletsane built a model tree forest for regression tasks [43], and an LMT forest was built to predict steel plate faults [41]. We also developed **the Enhanced Nearest Centroid Model Tree Forest (ENCMTF) classifier,** which is a forest of ENCMT classifiers.

There is not a single specific procedure to evaluate the quality of a new classifier. However, validity, reproducibility, comparability, and representativeness are important issues [44]. Stapor et al. [45] suggest that the authors of a classifier should not test the performance since they can be biased and suggest using "an independent classifier testing platform". Czmil et al. [17] built such software to enable the comparison of classifiers "to evaluate classification performance, reproducibility, and statistical reliability." This open-source software is called *cacp*, and we have used it extensively in our experiments. It is an easy-to-use platform, and it helped us to save a lot of time in our experiments. It also allows for the easy replication of any study. The use of this comparison software (*cacp* library) provides a rigid experimental design that brings validity to experimentation. It provides k-fold cross-validation and splitting into train and test datasets in a black-box manner without allowing manipulation of the comparison process. This guarantees that it is impossible to manipulate data to reach desired performance levels.

3 Methods

The ENC Algorithm is based on the NC algorithm, where we calculate the centroids of each class, and then while assigning a label to a test instance, we look for the nearest centroid. We make two major adjustments to the distance calculation for centroids. First, we increase the contributions of variables to distance calculation which have a higher predictive power (measured by a function of information value). Second, we increase the distance to the class where the variable shows less variability around its mean. The details are provided below but we first provide the formal definition of the binary classification problem and provide some preliminary information.

3.1 Problem definition

In the training dataset of a binary classification problem, we have n instances and m features. It means we have feature vectors $\boldsymbol{X} = (X_1, \ldots, X_n)^T \in \mathbb{R}^{nxm}$ and the target vector $\boldsymbol{Y} \in \mathbb{R}^n$ where each target Y_i is paired to a data point X_i and $Y_i \in \{0,1\}$. These two values of the target vector are also called class labels, so we have two classes, class 0 and class 1. Each feature vector is indexed with i and it corresponds to a row in the dataset. The feature is indexed with j and it corresponds to a column in the dataset.

The goal is to learn a function $f : \mathbb{R}^m \rightarrow \{0,1\}$ that maps feature vectors to class labels. In other words, we want to find a function that accurately predicts class labels from the corresponding input feature vectors. This function is called a predictive model. A predictive model is built on a training dataset using a predictive algorithm, and this model may be used to predict classes of new data points in test datasets.

Note that, in this study, we use the words variable, attribute, predictor, feature, and field interchangeably, and they all mean the same thing.

3.2 Information value

The predictive powers of features are computed by Information Value (IV), which is computed via Weight of Evidence values [4]. The Weight of Evidence (WoE) of a value of a categorical feature is computed as the logarithm of the ratio of the proportion of class 1 instances to the proportion of class 0 instances in that value. Therefore, highly negative values correspond to the high density of class 0, and highly positive values correspond to the high density of class 1. The formulation of the weight of evidence is given in Eq. 1. WoE values are calculated for each category c of the feature.

$$WoE_c = ln\left(\frac{proportion\ of\ class\ 1\ instances\ falling\ in\ category\ c}{proportion\ of\ class\ 0\ instances\ falling\ in\ category\ c}\right) \tag{1}$$

As given in Eq. 2, for a feature j with C distinct categories, information value (IV) is the weighted sum of the *weight of evidence* values where weights are calculated as the differences of percentage shares of classes.

$$IV_j = \sum_{c=1}^{C} (Proportion\ of\ class\ 1\ instances\ falling\ in\ category\ c - Proportion\ of\ class\ 0\ instances\ falling\ in\ category\ c) \bullet WoE_c \tag{2}$$

The predictive power of a feature is accepted as low if its information value is below 0.1, as medium if its information value is between 0.1 and 0.3, and as high if its information value is above 0.3. Even though it is possible to get big information values (even bigger than 1), it is advised to check the design against leakages if it is above 0.5 [4].

Since the Weight of Evidence can be calculated only for individual attribute values, Information Value is defined only for features with categorical values. Table 1 shows an example calculation for the "Education" feature. "Dist. Diff." column gives the result of the subtraction operation in Eq. 2, which acts as the weight of the corresponding WoE value. IV is calculated as the sum of IV parts, and IV parts are calculated for each categorical value of the feature by multiplying "Dist. Diff." with WoE.

We will consider using the IV of a feature as its weight in calculating the distance of instances to the centroids of the classes. In this regard, we can use it as is, or we can use it after a transformation. So, we define the weight w_j of a feature j as a generic function of information value, as in Eq. 3.

$$w_j = \mathrm{f}(IV_j) \tag{3}$$

Later, at the beginning of Sect. 5, we will talk about the performances of several alternative transformation functions such as square and logarithm.

3.3 Distance calculation and the ENC algorithm

First, we compute class centroids for each feature j for each class k, as given in Eq. 4:

$$\bar{x}_{jk} = \frac{1}{n} \sum_{i=1, Y_i=k}^{n} x_{ij}\ for\ all\ j, k \in \{0,1\} \tag{4}$$

Second, the information values are computed as given in Eq. 2, and the transformation in Eq. 3 is applied to get w_j for each feature j. Then we calculate within-class variance v_{jk} for each feature j and for each class k and compute distances to each class centroid by Eq. 5:

Table 1 An example of the calculation of the weight of evidence and information value

Education	count	Class 1	Class 0	Class 1%	Class 0%	WoE	Dist. Diff	IV part
Secondary School	3,000	600	2400	60	27	81%	33%	0.27
High School	3,000	300	2700	30	30	0%	0%	–
University	4,000	100	3900	10	43	– 147%	– 33%	0.49
Total	10,000	1000	9000	100	100		$IV_{Education}$	0.76

$$Distance\ of\ X_i to\ class\ k = D_{ik} = \sum_{j=1}^{m} \frac{w_j (x_{ij} - \bar{x}_{jk})^2}{v_{jk}} for k \in \{0,1\} \tag{5}$$

In the end, the predicted class is determined as the one with the smaller distance. We also calculate a similarity score to class 1, as given in Eq. 6.

$$Score\ for\ X_i = D_{i0} - D_{i1}\ for\ all\ i \tag{6}$$

The distance calculation in Eq. 5 considers the variance of the values of each feature in that class. Higher variance means it is more likely to have class members located far away from the center so that the distance could be computed as smaller. In other words, if the values of a feature are scattered throughout the class, being away from the center becomes less critical, so the distance calculation for that feature may get a smaller weight than a feature that has almost a constant value in that class. That is the intuition behind placing the variance to the denominator. Welch's t-test has a similar usage of variance in the calculation of its test statistics, and by dividing the information value to variance we are producing a kind of "information density" for each feature in both classes.

Before calculating variance, we first winsorize each feature by trimming each end. The reason for using winsorization is to remove outliers so that the centroid calculation becomes healthier, as outliers can easily shift the centroids. Via winsorization, the variance calculation also becomes more reliable.

Figure 1 shows the flowcharts for the training and scoring phases of the ENC algorithm. The pseudo-code of the algorithm is given next.

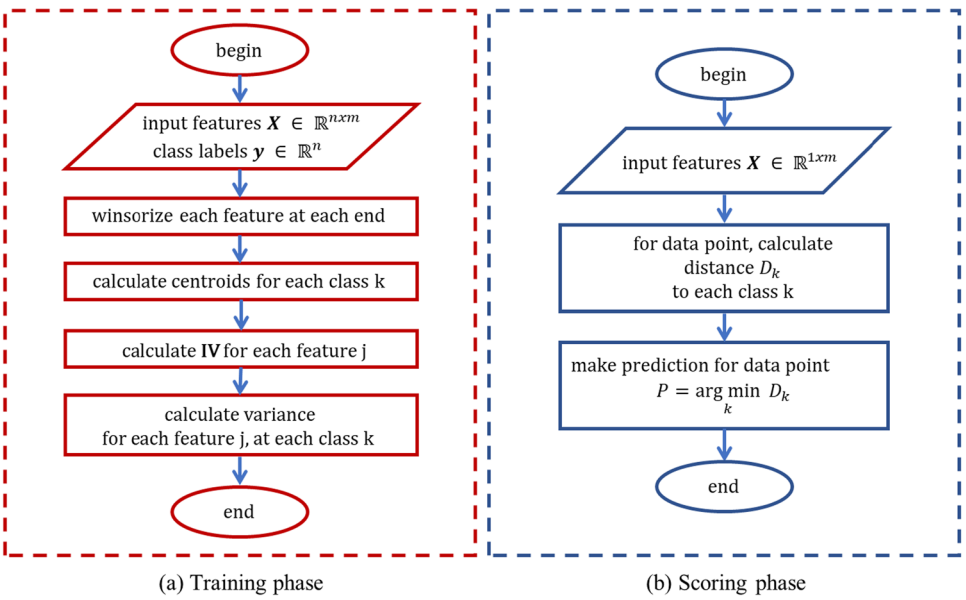

Fig. 1 Flowcharts for the training and scoring phases of ENC classifier

Algorithm 1. ENC Algorithm.

Learning Phase

Input: Training set D_{train}
Output: Information values of each feature, mean and variance of each feature at each class
Steps:

1. Winsorize each feature by 5% at each end
2. Compute the information value of each feature to predict the target variable at D_{train}
3. Compute centroids coordinate for each class
4. Compute the variance of each feature at each class

Prediction Phase

Input: Test set D_{test}
Output: Prediction scores for each data point $x_i \in D_{test}$
Steps:

1. **For** each $x_i \in D_{test}$ **do**
2. Compute the distance to each class
3. Calculate the final score as the difference of distances for class 1 and class 0
4. Determine the predicted class as having the smallest distance
5. **End For**

3.4 ENCMT algorithm

A Decision Tree algorithm is a non-parametric supervised learning method used for either classification or regression. A Decision Tree Classifier uses a tree-like structure to classify instances based on their feature values. Each node in the tree represents a set of instances, branches represent split decisions, and terminal nodes indicate the class label. The split decisions aim to produce homogeneous subsets of data in the new nodes. These decisions are made by calculating a split quality metric such as gini or entropy overall features. In our work, we chose gini as the split quality metric.

Model trees are a hybrid of the decision tree algorithm with another base algorithm, which is applied at terminal nodes. If this base algorithm is a logistic tree, then we have a Logistic Model Tree (LMT). Similar to LMT, we developed the Enhanced Nearest Centroid Model Tree (ENCMT) classifier, where the ENC classifier is used at each leaf of the decision tree. At the model training phase, first, a decision tree is built, and data is split into terminal nodes, i.e., leaves of the decision tree. At that point, we have s subsets of the dataset if we have s leaves, and we could build s ENC models for each subset of the training dataset. At the end of the training, for each subset, we have an ENC model with centroid, variance, and information value information. At the model scoring phase, first, we put the test data into the decision tree to have data points distributed among various leaves. Then, we run the corresponding ENC model for each leaf to have predictions for each data point. Figure 2 depicts the formation of the model tree. Figure 3 shows the flowcharts for the training and scoring phases of the ENCMT algorithm. The pseudo-code of this algorithm is given next.

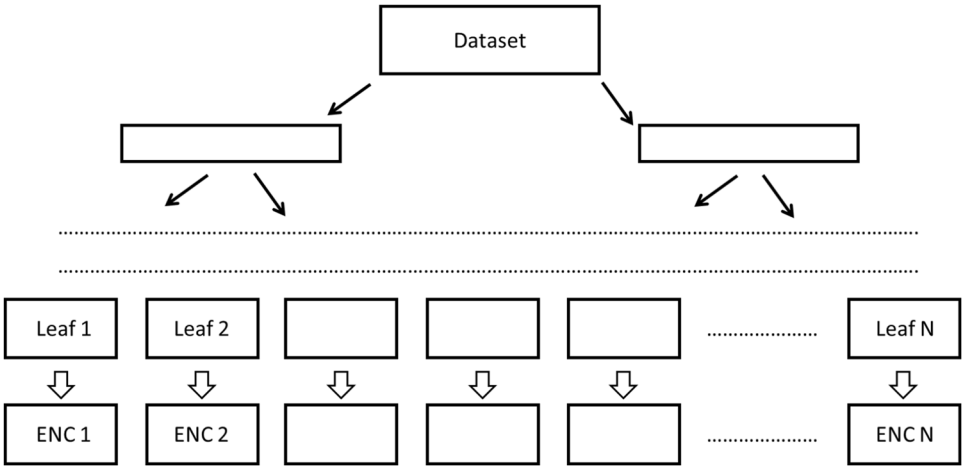

Fig. 2 Building ENCMT from Decision Tree and ENC Classifiers

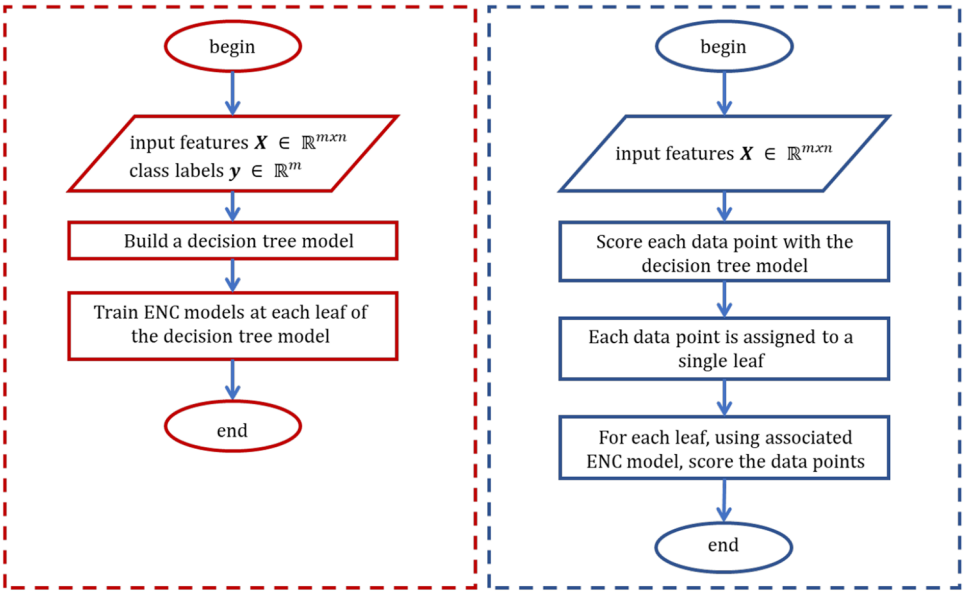

Fig. 3 Flowcharts for the training and scoring phases of ENCMT classifier

Algorithm 2. ENCMT Algorithm.

Learning Phase

Input: Training set D_{train}
Output: A decision tree model and a set of ENC models
Steps:

1. Build a decision tree via the Decision Tree Classifier
2. **For** each leaf of the decision tree, **do**
3. Build an ENC model via ENC Classifier
4. **End For**

Prediction Phase

Input: Test set D_{test}
Output: Prediction scores for each data point $x_i \in D_{test}$
Steps:

1. **For** each $x_i \in D_{test}$ **do**
2. Compute the corresponding leaf for that data point from the decision tree
3. Use the ENC model for that leaf to calculate the prediction
4. **End For**

3.5 ENCMTF algorithm

A random forest classifier is a bagging classifier made up of a set of independent decision trees that are ensembled to reach a final prediction. The randomness comes from bootstrap sampling made on rows of the training dataset, random feature selection, and the decision tree split decisions. This ensemble method improved the performance of the decision tree classifier and has been widely used for many years. Inspired by this, we also built a random forest using the Enhanced Nearest Centroid Model Tree (ENCMT) classifier instead of the Decision Tree classifier. It uses a random subset of rows and a random subset of features for each model tree. At the training phase of the forest, a set of ENCMT models are built, independent from each other. At the scoring phase of the forest, each data point is scored using each ENCMT model. Then, their prediction scores are averaged to reach the final prediction score. The predicted class is determined by comparing this final score with 0.5 (the default threshold). Figure 4 shows the flowcharts for the training and scoring phases of the ENCMTF algorithm. The pseudo-code of the algorithm is given next.

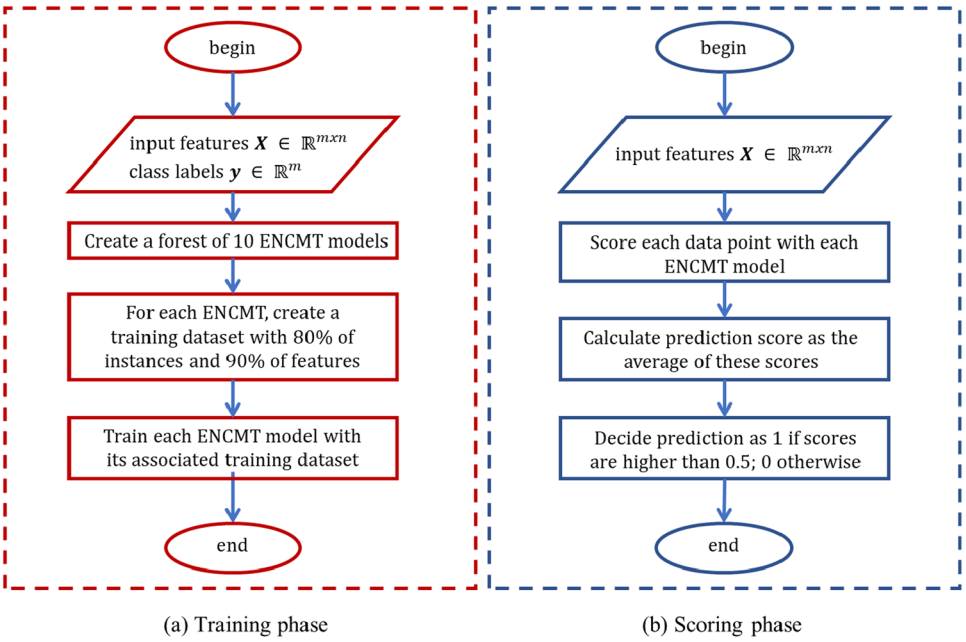

Fig. 4 Flowcharts for the training and scoring phases of ENCMTF classifier

Algorithm 3. ENCMTF Algorithm.

Learning Phase

Input: Training set D_{train}
Output: A set of ENCMT models (a set of decision tree models, each having a set of ENC models)
Steps:

5. Create a forest with ENCMT models
6. **For** each ENCMT of the forest, **do**
7. Create 80% sample of the training data set rows
8. Create a 90% sample of the training data set features
9. **For** each leaf of the decision tree, **do**
10. Build an ENC model via ENC Classifier
11. **End For**
12. **End For**

Prediction Phase

Input: Test set D_{test}
Output: Prediction scores for each data point $x_i \in D_{test}$
Steps:

1. **For** each $x_i \in D_{test}$ **do**
2. **For** each ENCMT of the forest
3. Take only features used at that ENCMT
4. Compute the corresponding leaf for that data point from the decision tree
5. Use the ENC model for that leaf to calculate the prediction
6. **End For**
7. Take the average of all probabilities produced from the forest trees
8. **End For**

Figure 5 shows the ROC curves of ENC, ENCMT and ENCMTF on a sample dataset. In this example ENCMT was good at the first percentiles but later performed poorly. On the other hand, ENCMTF performed better than the others and its curve is much more balanced (Fig. 5).

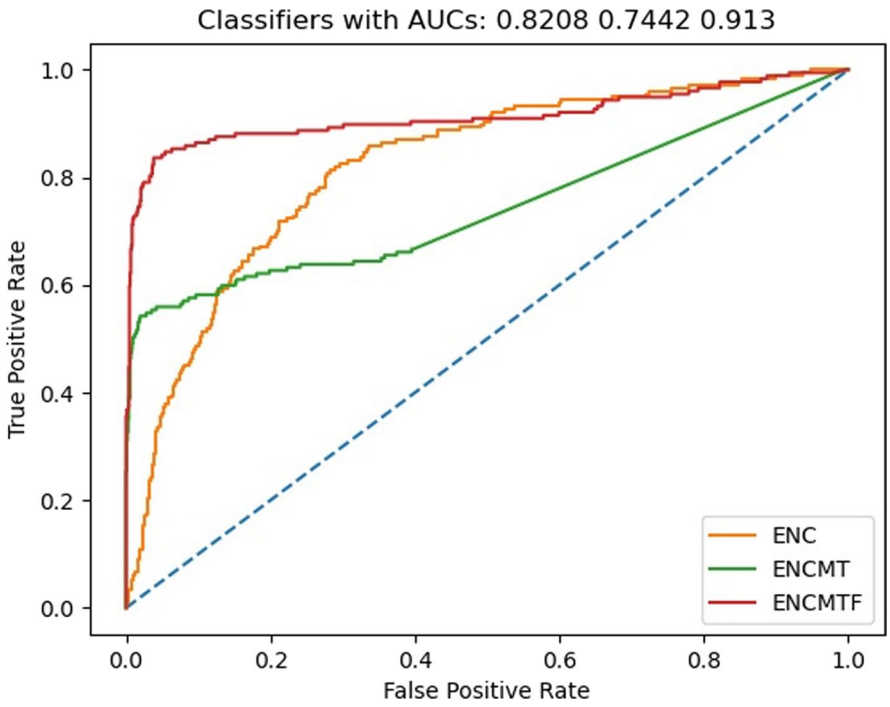

Fig. 5 ROC curves of three classifiers on a sample dataset

These three algorithms do not require scaling (such as standardization or normalization) but scaling has the potential to improve performance. They require that there is no missing value in the dataset but they are good at sparse data since the information value effectively calculates the impact of rare categories.

4 Experimental procedure

4.1 Experimental setup

All experiments are performed on a PC having 32 GB RAM and 4 Core Intel i7 11th Gen CPU's running at 2.80 GHz. This PC has Microsoft Windows 11 Pro, Python 3.12.7, Jupyter-notebook 7.2.2, Conda 24.11.0, and Anaconda 3.

4.2 Coding and experimentation details

The ENC and ENCMT algorithms are implemented as a class in Python. They both inherit the base estimator of the Scikit-Learn library [46], and they have "fit", "predict," and "predict_proba" methods, so it is Scikit-Learn compatible. This was also a requirement to use it in the cacp library.

The cacp library makes fivefold or tenfold cross-validation, and we used the default option of tenfold. To enable the reproducibility of results, we provided a seed (random_state) value of 42 to the cacp library and also to the benchmarked classifiers if they have such input.

Since continuous numeric features should be discretized for the weight of evidence calculation, we created 10 bins of equal size for numeric features. For winsorization, we preferred to trim 5% at each end of the training dataset, i.e., values below five percentile points are raised to 5 percentile points, and values above 95 percentile points are trimmed to 95 percentile points.

4.3 Datasets used

We used 30 datasets in the comparison software that belong to binary classification problems. These datasets originated from the KEEL repository [47], and they have been frequently used in the literature. Table 2 lists the datasets used together with their characteristics (first four columns). As can be seen in this table, there is a great diversity in this experimental

Table 2 The binary datasets used for comparison and AUC values of algorithms

Dataset	Instances	Features	Minority Class %	ENC	DT	LMT	ENCMT	RF	LMTF	ENCMTF	Average
Appendicitis	106	7	**20%**	**0.81**	0.50	0.68	0.71	0.50	0.63	0.73	0.64
Australian	690	14	44%	0.85	0.84	0.84	0.81	**0.87**	0.86	0.85	0.85
Banana	5,300	2	45%	0.50	0.88	0.89	0.85	0.89	**0.89**	0.88	0.82
Bands	365	19	37%	0.50	0.60	0.61	0.64	0.58	0.60	**0.66**	0.59
Breast	277	9	**29%**	0.64	**0.66**	0.49	0.65	0.64	0.38	0.63	0.58
Bupa	345	6	42%	0.50	0.65	0.68	0.67	0.64	**0.71**	0.68	0.64
Chess	3,196	36	48%	0.50	0.96	0.98	0.98	0.97	0.98	**0.98**	0.89
Crx	653	15	45%	0.87	0.86	0.85	0.79	**0.87**	0.61	0.84	0.81
German	1,000	20	**30%**	0.51	0.66	0.61	0.65	0.63	**0.68**	0.65	0.62
Haberman	306	3	**26%**	0.50	**0.60**	0.60	0.57	0.55	0.60	0.58	0.57
Heart	270	13	44%	0.57	0.73	0.80	0.80	0.75	**0.83**	0.81	0.75
Hepatitis	80	19	**16%**	0.50	0.50	0.72	**0.87**	0.50	0.68	0.86	0.63
Housevotes	232	16	47%	0.97	**0.97**	0.97	0.80	**0.97**	**0.97**	0.91	0.94
Ionosphere	351	33	36%	0.50	0.88	0.88	0.89	0.88	0.89	**0.90**	0.82
Magic	19,020	10	35%	0.58	0.81	0.66	0.79	0.83	0.33	**0.83**	0.67
Mammographic	830	5	49%	0.73	0.84	0.83	0.78	0.83	**0.85**	0.80	0.81
monk-2	432	6	47%	0.50	0.97	0.98	0.98	0.97	0.98	**0.98**	0.90
Mushroom	5,644	22	38%	0.50	0.99	**1.00**	**1.00**	0.99	0.90	**1.00**	0.90
Phoneme	5,404	5	**29%**	0.73	0.78	0.81	0.79	0.81	**0.83**	0.82	0.79
Pima	768	8	35%	0.61	0.70	0.71	0.71	0.70	0.72	**0.72**	0.69
Ring	7,400	20	50%	0.50	0.87	0.69	0.88	0.91	0.73	**0.92**	0.76
Saheart	462	9	35%	0.61	0.62	0.65	0.64	0.65	**0.65**	0.64	0.64
sonar	208	60	47%	0.50	0.71	**0.78**	0.76	0.71	0.77	0.74	0.71
spambase	4,597	57	39%	0.63	0.89	0.92	0.89	0.91	0.65	**0.93**	0.82
spectfheart	267	44	**21%**	**0.68**	0.52	0.62	0.65	0.50	0.65	0.67	0.60
tic-tac-toe	958	9	35%	0.50	0.65	0.74	0.75	0.69	0.79	**0.89**	0.69
Titanic	2,201	3	**32%**	0.50	0.69	**0.71**	0.54	0.69	**0.71**	0.54	0.64
twonorm	7,400	20	50%	0.50	0.80	0.74	0.88	0.91	0.77	**0.95**	0.77
Wdbc	569	30	37%	0.82	0.90	0.95	0.94	0.89	**0.95**	0.95	0.91
wisconsin	683	9	35%	0.97	0.93	0.96	0.96	0.92	**0.97**	0.96	0.95
Averages				**0.62**	**0.77**	**0.78**	**0.79**	**0.77**	**0.75**	**0.81**	0.75
Being best				**2**	**3**	**3**	**2**	**3**	**11**	**11**	

(*) Minority class %s are in bold if there is a class imbalance (less share than 1/3)

(**) The highest AUC values are in bold

framework with different numbers of features. The table also lists the AUC values of algorithms being experimented for each dataset.

5 Results

As defined in Eq. 3, a function of information value is used as weights in distance calculation. We tested different transformations of information value on the performance of the ENC classification algorithm. Since the information value is mostly below one and close to zero, its logarithm becomes negative, which is not meaningful, so adding 1 was a good starting point. Also, taking its square and square root were good options. In our experiments $(1 + IV)^2$ turned out to be the best one, and we used this transformation for the transformation of information value in our implementation.

We compared ENCMT and ENCMTF with other algorithms pairwise using cacp software over 30 datasets. Since cacp software makes tenfold cross-validation, 300 models are built for each algorithm. Table 2 lists the average AUC metric of 10 models produced by tenfold cross-validation for each dataset. AUC metric is the area under the Receiver Operating

Characteristics (ROC) curve. Since it is independent of decision cut-off, it is a global measure of classifier performance and it is chosen for comparisons. On average, ENCMTF is the best classifier, whereas ENCMT is the second-best one. The average AUC performance of LMT drops from 0.78 to 0.75 in its bagging version (LMTF). However, LMTF is also a good classifier since it is the best algorithm for 11 datasets. Similar to LMTF, ENCMTF is the best classifier for 11 datasets. In Table 2, we can also observe that the average performance of the algorithms is worse for problems with higher imbalances.

ENCMT is a hybrid algorithm using both ENC and DT. The comparison of its performance with ENC and DT showed clear progress in terms of AUC. We also compared its performance with LMT, and it performed better. On the other hand, its bagging derivative, ENCMTF, could be compared with RF and LMTF. When we did that, we saw better performance in terms of AUC.

There are other performance metrics to evaluate the performance of a classifier. The cacp software calculates Accuracy, Precision, Recall, and F1 scores in addition to AUC. Accuracy is the fraction of correctly classified instances; precision is the ratio of correct classifications among positive predictions; recall is the ratio of correctly classified positive instances among all positive instances; and F1 score is the harmonic mean of precision and recall. Table 3 lists the average values of these metrics over 30 datasets for each algorithm. On average ENCMT is a good classifier surpassing even its bagging rivals (RF and LMTF) in terms of AUC. Its bagging variant, ENCMTF competes with the state-of-the-art classifiers LightGBM and XGBoost. Next, we compare the algorithms pair-wise with statistical tests.

Using cacp software, we first compared ENCMT with ENC and found that its AUC is better than ENC at 24 datasets (out of 30). The averages of performance metrics are listed in Table 4. The average performance of ENCMT is better than ENC in all 5 of the performance metrics. We also checked their significance via the Wilcoxon signed-rank tests and found that p-values are less than 0.0001 for all 5 of the performance metrics. This proves the idea that instead of having a single set of weights in ENC, the set of weights should differ from subset to subset and this subset operation is successfully carried by the decision tree algorithm.

Second, we compared ENCMT with the Decision Tree (DT) classifier and found that its AUC is better than DT at 19 datasets (out of 30). The averages of performance metrics are listed in Table 5. The average performance of ENCMT is better than DT in all 5 of the performance metrics. We also checked their significance via the Wilcoxon signed-rank tests and found that p-values are less than 0.01 for AUC, precision and F1 score, and about 0.12 for accuracy and recall. Having

Table 3 Summary of comparison results

Algorithm	AUC	Accuracy	Precision	Recall	F1
ENC	0.6191	0.6043	0.5848	0.6043	0.5186
DT	0.7655	0.8213	0.8074	0.8213	0.8071
LMT	0.7778	0.8159	0.8141	0.8159	0.8103
ENCMT	0.7867	0.8234	0.8272	0.8234	0.8145
RF	0.7712	0.8325	0.8167	0.8325	0.8152
LMTF	0.7526	0.7944	0.7912	0.7944	0.7870
ENCMTF	0.8100	0.8489	0.8474	0.8489	0.8398
LightGBM	0.8380	0.8704	0.8702	0.8704	0.8652
XGBoost	0.8314	0.8635	0.8633	0.8635	0.8589
Averages	0.7725	0.8083	0.8025	0.8083	0.7907

Table 4 Comparison of ENCMT with ENC

Algorithm	AUC	Accuracy	Precision	Recall	F1
ENCMT	0.7867	0.8234	0.8272	0.8234	0.8145
ENC	0.6191	0.6043	0.5848	0.6043	0.5186
p-values	0.0000	0.0000	0.0000	0.0000	0.0000

Table 5 Comparison of ENCMT with DT

Algorithm	AUC	Accuracy	Precision	Recall	F1
ENCMT	0.7867	0.8234	0.8272	0.8234	0.8145
DT	0.7655	0.8213	0.8074	0.8213	0.8071
p-values	0.0059	0.1189	0.0003	0.1189	0.0032

the performance of ENCMT as better than DT, we conclude that the score differentiation via ENC within each leaf of the decision tree is successful. In other words, ENCMT successfully removed the leaf-based fixed score obstacle of DT.

Third, we compared ENCMT with the Logistic Model Tree (LMT) classifier and found that its AUC is better than LMT at 15 datasets (out of 30). The averages of performance metrics are listed in Table 6. The average performance of ENCMT is better than LMT in all 5 of the performance metrics. We also checked their significance via the Wilcoxon signed-rank tests and found that all p-values are less than 0.001. These results show that ENCMT is significantly better than LMT which is a widely used algorithm.

Fourth, we compared the ENCMT Forest (ENCMTF) Classifier with the Random Forest (RF) Classifier and found that its AUC is better than RF at 22 datasets (out of 30). The averages of performance metrics are listed in Table 7. The average performance of ENCMTF is better than RF in AUC and Precision metrics. We also checked their significance via the Wilcoxon signed-rank tests and found that p-values are less than 0.0001. ENCMTF showed significantly superior performance against RF which was an expected result since ENCMT was better than DT. Bagging with random feature and instance selection mechanisms did not harm the performance of ENCMT. Instead, they improved the performance, for example AUC increased to 0.8100 from 0.7867.

Fifth, we compared the ENCMT Forest (ENCMTF) Classifier with the Logistic Model Tree Forest (LMTF) Classifier and found that its AUC is better than RF at 15 datasets (out of 30). LMTF Classifier is a random forest with LMTs instead of decision trees. The averages of performance metrics are listed in Table 8. The average performance of ENCMTF seems better than LMTF in all 5 of the performance metrics. However, we also checked their significance via the Wilcoxon signed-rank tests and found that p-values are very high. ENCMTF also showed better performance against LMTF which was an expected result since ENCMT was better than LMT.

Lastly, we compared the ENCMT (ENCMT) Classifier with LightGBM and XGBoost classifiers and found that its AUC values are better at 5 datasets (out of 30). LightGBM and XGBoost are better on average (as seen at Table 3) but it is a good idea to give a chance to ENCMT since it is better at about %17 of the problems. Similar values were computed when we compared the ENCMTF Classifier with LightGBM and XGBoost. Compared to these two boosting algorithms, our algorithms, ENCMT and ENCMTF, are at their initial research steps.

In summary, ENCMT is better than both ENC and DT, which are the two algorithms on which it relies on. It also has better performance than LMT, which is a similar model tree algorithm. Finally, we showed that forests made up of EMCMT are better than Random Forest and have comparable performance against forests made up of LMT.

Table 9 lists the training and scoring times of benchmarked algorithms over 30 datasets. ENC is fast but ENCMT is slower since it is made up of many ENC's. Similarly, ENCMTF is very slow. This table gives an idea about scalability, but these figures could change as we change some implementation details such as using "numpy" library instead of "pandas" library and writing parallel coding to enable using multiple CPU cores. The run time of ENCMT could be divided into two phases: first, the time needed for decision tree, and second the time needed for each ENC model at each leaf. Both the

Table 6 Comparison of ENCMT with LMT

Algorithm	AUC	Accuracy	Precision	Recall	F1
ENCMT	0.7867	0.8234	0.8272	0.8234	0.8145
LMT	0.7778	0.8159	0.8141	0.8159	0.8103
p-values	0.0002	0.0003	0.0023	0.0003	0.0001

Table 7 Comparison of ENCMTF with RF

Algorithm	AUC	Accuracy	Precision	Recall	F1
ENCMTF	0.8100	0.8489	0.8474	0.8489	0.8398
RF	0.7712	0.8325	0.8167	0.8325	0.8152
p-values	0.0000	0.0000	0.0000	0.0000	0.0000

Table 8 Comparison of ENCMTF with LMTF

Algorithm	AUC	Accuracy	Precision	Recall	F1
ENCMTF	0.8097	0.8484	0.8475	0.8484	0.8395
LMTF	0.7526	0.7944	0.7912	0.7944	0.7870
p-values	0.8633	0.9194	0.7588	0.9194	0.8036

Table 9 Training and scoring times of algorithms

Algorithm	Train times	Prediction times
DT	0.0301	0.0003
RF	0.2174	0.0188
ENC	0.2477	0.0042
XGBoost	0.4868	0.0014
LightGBM	0.5455	0.0017
ENCMT	4.2159	0.1163
ENCMTF	33.5650	0.7442
LMT	300.2481	300.0129
LMTF	634.9392	633.5650

training and scoring runtimes of DT is small and the main effect comes from ENC models and their total time linearly increases with the number of leaves. At the training of a decision tree, it is possible to give the maximum number of leaves as an input parameter which is mostly used for preventing overfitting. By setting the maximum number of leaves parameter, it is possible to create a smaller tree and reduce the number of leaves. This will also proportionally reduce the ENCMT models' training and scoring times.

6 Conclusion and further work

6.1 Conclusion

In this research, a new general-purpose binary classifier is proposed. The ENCMT is born from two basic ideas:

1. At similarity calculations each feature should be weighted,
2. These weights should be different at different subsets of the data.

The first idea is answered by the ENC algorithm, which calculates weights as information value normalized by the variance. The second idea, which needs the partitioning of the data is accomplished by Decision Trees. The combination of these two ideas resulted a successful algorithm. We also used various clustering algorithms but none of them was successful. Unlike clustering algorithms, Decision Tree classifier is aware of the target variable and partitions data to maximize purity, which is suitable for ENC.

The ENCMT Classifier is a model tree algorithm that is explainable. It requires low memory since only class and feature-based information needs to be stored. It is simple, but its performance is better than the two algorithms it is made up of, ENC and DT. Its performance is better than that of a similar model tree algorithm, namely the Logistic Model Tree (LMT) Classifier.

The idea of the model tree could also be applied to bagging and boosting algorithms. We used ENCMT to build a random forest (ENCMTF) and showed that using ENCMTs instead of Decision Trees improved the performance.

Using ENC in each leaf removes the fixed score obstacle of decision trees and allows different scores within the leaves. Interpretability is an essential issue in machine learning applications. ENCMT partitions the data using a traditional univariate decision tree, which makes the partitioning easy to understand. Each leaf is an ENC classifier, which is a simple classifier and can also be easily understood.

Compared to recent advancements in deep learning, ENCMT does not require large training datasets and it is much more explainable.

This study has the following limitations:

1. The behavior of ENC, ENCMT and ENCMTF on imbalanced datasets are not analyzed. Such a work should use enough number of imbalanced datasets to be precise.
2. Since model trees are based on Decision Trees, all hyperparameters of the Decision Trees are also hyperparameters of ENCMT and may be tuned. Similarly, all hyperparameters of the Random Forest are also valid for ENCMTF. The effect of tuning these parameters on algorithm performance needs additional experimentation. These experiments could

also provide the sensitivity of algorithms to their parameters. Such work could utilize novel optimization approaches such as the Greylag Goose Optimization algorithm [48] and the Football Optimization Algorithm [49].

ENC's, ENCMT's, and ENCMTF's codes are publicly shared to encourage researchers in this field to further research.

6.2 Further work

Further work is possible by applying the following approaches:

1. A multiclass version of ENC could be developed, which leads to multiclass versions of ENCMT and ENCMTF. Since information value is valid only for two classes, this could start from aggregating information values of different classes with respect to a base class such as the majority class.
2. A similar approach might be applied to kNN, i.e., the features could be weighted by their predictive power at the distance calculation of neighbors, and this new kNN algorithm could be used as a base classifier at model trees.
3. ENCMT could be used as a base classifier for boosting algorithms.
4. ENC may be used to make split decisions when the tree is grown. The growth of the tree may be optimized to allow ENC to perform better in each leaf.
5. Instead of a fixed transformation of information value, different transformations could be implemented and selecting the transformation could be a new parameter of the algorithm.
6. Permutation feature importance values could be calculated at training to enhance interpretability.
7. With its interpretability advantage ENCMT could be used at semi-supervised learning, transfer learning or federated learning contexts.

Acknowledgements The authors declare that no funds, grants, or other support were received during the preparation of this manuscript. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author contributions MHO: Conceptualization, Investigation, Methodology, Software, Writing—original draft preparation, Validation. ED: Writing—review and editing, Supervision. SB: Writing—review, Supervision. SB: Writing—review, Supervision.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability The 30 datasets analyzed during the current study are available in the cacp library. Detailed information is available at https://cacp.readthedocs.io/en/latest/. They are also publicly available at the KEEL repository at https://sci2s.ugr.es/keel/datasets.php.

Code availability The code developed in this study is publicly available on GitHub at the address https://github.com/mhozcelik/ENCMT.

Declarations

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Tan PN, Steinbach M, Karpatne A, Kumar V, Introduction to data mining, 2nd Edition, 2020, Pearson Addison Wesley.
2. Duman E. Social media analytical CRM: a case study in a bank. JIFS. 2023;44(2):2631–42.
3. Mohri M, Rostamizadeh A, Talwalkar A. Foundations of machine learning. MIT press, 2018 ISBN 9780262018258.
4. Siddiqi N. Intelligent credit scoring: Building and implementing better credit risk scorecards. John Wiley & Sons, 2017., pp.186–197. https://doi.org/10.1002/9781119282396
5. Breiman L, Friedman JH, Olshen RA, Stone CJ. Classification and regression trees. New York: Chapman & Hall/CRC; 1984.
6. Kass G. An exploratory technique for investigating large quantities of categorical data. Appl Stat. 1980;29(2):119–27.
7. Quinlan JR. Decision trees and decision making. IEEE Trans Syst Man Cybern. 1990;20(2):339–46.
8. Quinlan JR. Learning with continuous classes. Mach Learn. 1992;1:81–106.
9. Larochelle H, Bengio Y, Louradour J, Lamblin P. Exploring strategies for training deep neural networks. J Mach Learn Res. 2009;10:1.
10. Witten IH, Frank E. Data mining: practical machine learning tools and techniques with Java implementations. ACM SIGMOD Rec. 2002;31(1):76–7.
11. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SigKDD international conference on knowledge discovery and data mining, pp. 785–794. 2016.
12. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY. Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Info Process Syst 2017;30
13. Januschowski T, Wang Y, Torkkola K, Erkkilä T, Hasson H, Gasthaus J. Forecasting with trees. Int J Forecast. 2022;38(4):1473–81.
14. Prasad BS, Gupta S, Borah N, Dineshkumar R, Lautre HK, Mouleswararao B. Predicting diabetes with multivariate analysis an innovative KNN-based classifier approach. Prev Med. 2023;174: 107619.
15. Hastie T, Tibshirani R, Friedman JH. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009, p.670.
16. Özçelik MH, Bulkan S. Nearest centroid classifier based on information value and homogeneity. In: Şen, Z., Uygun, Ö., Erden, C. (eds) Advances in intelligent manufacturing and service system informatics. IMSS 2023. Lecture Notes in Mechanical Engineering. Springer, Singapore. 2024. https://doi.org/10.1007/978-981-99-6062-0_5
17. Czmil S, Kluska J, Czmil A. CACP: classification algorithms comparison pipeline. SoftwareX. 2022;19: 101134.
18. Fawcett T. An introduction to ROC analysis. Pattern Recogn Lett. 2006;27(8):861–74.
19. Thulasidas M. Nearest centroid: A bridge between statistics and machine learning. In: 2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 9–16. IEEE, 2020. https://doi.org/10.1109/tale48869.2020.9368396
20. Setiawan B, Djanali S, Ahmad T. A study on intrusion detection using centroid-based classification. Procedia Comp Sci. 2017;124:672–81.
21. Raikar MM, Meena SM, Mulla MM, Shetti NS, Karanandi M. Data traffic classification in software defined networks (SDN) using supervised-learning. Procedia Comp Sci. 2020;171:2750–9.
22. Johri S, Debnath S, Mocherla A, Singk A, Prakash A, Kim J, Kerenidis I. Nearest centroid classification on a trapped ion quantum computer. NPJ Quantum Inf. 2021;7(1):122.
23. Tibshirani R, Hastie T, Narasimhan B, Chu G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. Proc Natl Acad Sci. 2002;99(10):6567–72.
24. Ren S, Mai Q. The robust nearest shrunken centroids classifier for high-dimensional heavy-tailed data. Electron J Stat. 2022;16(1):3343–84.
25. Sahtout MO, Wang H, Ghimire S. Different thresholding methods on Nearest Shrunken Centroid algorithm. Commun Stat B Simul Comput. 2024;53(3):1444–60.
26. Ren S, Yang M, Mai Q. Decorrelated nearest shrunken centroids for tensor data. Stat. 2024;13(3): e720.
27. Fraiman N, Li Z. Classification with nearest disjoint centroids. arXiv preprint arXiv:2109.10436. 2021.
28. Wang W, Han C, Zhou T, Liu D. Visual recognition with deep nearest centroids. arXiv preprint arXiv:2209.07383. 2022.
29. Xie K, Hou Y, Zhou X. Deep centroid: a general deep cascade classifier for biomedical omics data classification. Bioinformatics. 2024;40(2):btae039.
30. Elen A, Avuçlu E. Standardized variable distances: a distance-based machine learning method. Appl Soft Comput. 2021;98: 106855.
31. Fan L, Ding Y. Research on risk scorecard of sick building syndrome based on machine learning. Build Environ. 2022;211: 108710.
32. Niño-Adan I, Manjarres D, Landa-Torres I, Portillo E. Feature weighting methods: a review. Expert Syst Appl. 2021;184: 115424.
33. Cunningham P, Delany SJ. k-Nearest neighbour classifiers-a Tutorial. ACM Comput Surv (CSUR). 2021;54(6):1–25.
34. Frank E, Wang Y, Inglis S, Holmes G, Witten IH. Using model trees for classification. Mach Learn. 1998;32:63–76.
35. Kohavi R. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In Kdd. 1996;96:202–7.
36. Landwehr N, Hall M, Frank E. Logistic model trees. Mach Learn. 2005;59:161–205.
37. Costa VG, Pedreira CE. Recent advances in decision trees: an updated survey. Artif Intell Rev. 2023;56(5):4765–800.
38. Zhou X, Yan D. Model tree pruning. Int J Mach Learn Cybern. 2019;10:3431–44.
39. Zhou X, Chen S, Peng N, Zhou X, Wang X. Uncertainty guided pruning of classification model tree. Knowl-Based Syst. 2023;259: 110067.
40. Plati DK, Tripoliti EE, Bechlioulis A, Rammos A, Dimou I, Lakkas L, Watson C, McDonald K, Ledwidge M, Pharithi R, Gallagher J. A machine learning approach for chronic heart failure diagnosis. Diagnostics. 2021;11(10):1863.
41. Ghasemkhani B, Yilmaz R, Birant D, Kut RA. Logistic model tree forest for steel plates faults prediction. Machines. 2023;11(7):679.
42. Ghasemkhani B, Balbal KF, Birant D. A new predictive method for classification tasks in machine learning: Multi-class multi-label logistic model tree (MMLMT). Mathematics. 2024;12(18):2825.
43. Moletsane, Phenyo Phemelo, Model Tree Forests, Dissertation, University of Pretoria, 2019. https://engel.pages.cs.sun.ac.za/files/phenyoMoletsane.pdf. Accessed 10 Nov 2024.
44. Hoffmann F, Bertram T, Mikut R, Reischl M, Nelles O. Benchmarking in classification and regression. WIREs Data Mining Knowl Discov. 2019;9(5): e1318.

45. Stapor K, Ksieniewicz P, García S, Woźniak M. How to design the fair experimental classifier evaluation. Appl Soft Comput. 2021;104: 107219. https://doi.org/10.1016/j.asoc.2021.107219.
46. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J. Scikit-learn: machine learning in Python. JMLR. 2011;12:2825–30.
47. Alcalá-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F. KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multiple-Valued Logic Soft Comput. 2011;17(2–3):255–87.
48. El-Kenawy E-S, Khodadadi N, Mirjalili S, Abdelhamid AA, Eid MM, Ibrahim A. Greylag goose optimization: nature-inspired optimization algorithm. Expert Syst Appl. 2024;238: 122147.
49. El-Kenawy ES, Rizk FH, Zaki AM, Mohamed ME, Ibrahim A, Abdelhamid AA, Khodadadi N, Almetwally EM, Eid MM. Football optimization algorithm (FbOA): a novel metaheuristic inspired by team strategy dynamics. JAIM. 2024;8(1):21–38.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.