**TEL AVIV UNIVERSITY**

IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING   
The Zandman-Slaner Graduate School of Engineering

**A Decision-Trees Based Approach for Ordinal Classification Problems**

By

**Matan Marudi**

A thesis submitted towards the degree of

Master of Science in Industrial Engineering

February 2021

**TEL AVIV UNIVERSITY**

IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING   
The Zandman-Slaner Graduate School of Engineering

**A Decision-Trees Based Approach for Ordinal Classification Problems**

By

**Matan Marudi**

This research was carried out at Tel Aviv University

in the Department of Industrial Engineering

Faculty of Engineering under the supervision of Dr. Gonen Singer

and Prof. Irad Ben-Gal

February 2021

**Acknowledgements**

I would like to thank my beloved wife Natali, for supporting me throughout all the years. You are my rock, without you none of this would have happened. You have been there through thick and thin, and I am forever grateful for that.

I feel honored and grateful for my two advisors, Prof. Irad Ben-Gal and Dr. Gonen Singer for giving me the opportunity to fulfill my professional and personal goals and aspirations. Prof. Irad Ben-Gal and Dr. Gonen Singer have been supporting me throughout the whole process, giving me the confidence, support and freedom needed to complete this research successfully. This amazing achievement would have never happened without the incredible presence of these two great minds.

I would also like to thank my parents, Avi and Zvia, who have always believed in me and pushed me to excellence. My dear parents have encouraged me to pursue my dreams from day one, and this accomplishment is just as much theirs, as it is mine.

Last but not least, I would like to dedicate this paper to my unborn daughter. I wish my efforts and achievements will set a personal example for you, teaching you that with hard work and dedication, everything is possible - the sky is the limit!

This page was intentionally left blank.

**Abstract**

In this research, a novel approach for ordinal classification problems is proposed while aiming to predict a predefined objective class, which is based on modification of the splitting criteria of decision trees’ algorithms. The proposed approach takes into consideration the nature ordinal of the data, as well as the magnitude of the potential classification error from a predefined predicted objective. It is shown how the proposed approach can be used during the creation of a tree-model, and how it can be extended and used for decision tree based boosting and bagging algorithms. In order to improve the classification performance, an ensemble-based scheme was used to combine different ordinal and non-ordinal classifiers. To evaluate the proposed approach, several known ordinal data sets were used, which were studied and evaluated in previous research studies, and it was found that the proposed ordinal approach significantly outperforms its counterpart non-ordinal models and state-of-the-art methods. The results are validated over various performance measures which are commonly used for ordinal and non-ordinal classification problems.

This page was intentionally left blank.

**Table of Contents**

List of Relevant Notations and Terminology viii

List of Figures ix

List of Tables x

1 Introduction 11

2 Background and Related Work 14

3 Ordinal decision tree based algorithms 19

3.1 Objective Based Information Gain (OBIG) measure 19

3.2 Implementing of OBIG measure in Ordinal decision tree 21

4 Ensemble based Ordinal algorithms 23

4.1 Ordinal Random Forest algorithm 23

4.2 Ordinal Boosting algorithm 25

4.3 Majority voting approach 28

5 Experiment 29

5.1 Data description 29

5.2 Performance criteria and evaluation method 30

5.3 Explanation of experimental stages and models used 31

33

6 Results and Discussion 33

6.1 Results of ordinal decision tree-based algorithms vs. non-ordinal counterparts. 33

6.2 Comparison of the proposed ordinal decision tree-based models results with previous published results 45

6.3 Results of ordinal decision tree-based models compared to state-of-the-art models 51

6.4 Majority voting results based on non-ordinal and ordinal models 52

7 Conclusion & Future Work 54

References 56

# List of Relevant Notations and Terminology

|  |  |
| --- | --- |
| **Notation** | **Description** |
|  | Data set |
|  | The number of samples in data set , |
|  | The sample in data set |
|  | Number of possible ordered predicted values |
|  | The predicted value in data set |
|  | The number of features in data set |
|  | The value of the feature in m sample |
|  | Attribute |
|  | Group of features { |
|  | The ordinal class ranking set where denote the lowest ranking and the highest ranking |
|  | A function that returns the value of class |
|  | Statistic function calculated based on samples of labels given the value function |
|  | The probability of class |
|  | The absolute deviation of the value of class from , |
|  | A normalization factor that smooths the weights’ distribution over the different classes |
|  | Objectives based entropy calculated based on statistic , predicted values and class values |
|  | Objective-based information gained from attribute |
| *L* | Number of classifiers in an ensemble model |
|  | Classifier in an ensemble model |
|  | Subset of the data that classifier trained on |
|  | Subset of the feature’s classifier trained on |
|  | The weight of sample in iteration in a bagging model |
|  | The total errors of the classifier during the boosting process |
|  | The error of the sample using the classifier during the boosting process |
|  | The weight of the classifier after evaluating |
|  | Normalization factor |
|  | Number of samples that are tied on the true class |
|  | Number of samples that are tied on the estimated class |
|  | Kendall’s correlation coeﬃcient |

# List of Figures

[Figure 1: The meta-code of the proposed ordinal decision tree classifier based on the OBIG measure. 22](#_Toc63114028)

[Figure 2 The meta-code of the proposed ordinal random-forest classifier based on the OBIG measure. 24](#_Toc63114029)

[Figure 3 The meta-code of the proposed ordinal AdaBoost classifier based on the OBIG measure 27](#_Toc63114030)

[Figure 4 Pseudo code for majority voting approach based on *L* decision tree-based algorithms 28](#_Toc63114031)

[Figure 5 AUC relative decreasing score per data set non-ordinal models 40](#_Toc63114032)

[Figure 6 AUC relative decreasing score per data set ordinal models 41](#_Toc63114033)

[Figure 7 AUC score for 3,5,10 class for Random Forest model and Mode statistic over benchmark data sets 42](#_Toc63114034)

[Figure 8 AUC score for 3,5,10 classes for decision tree model and Mode statistic over benchmark data sets 42](#_Toc63114035)

[Figure 9 AUC score for 3,5,10 classes for AdaBoost with ordinal constructor and Mode statistic over benchmark data sets 43](#_Toc63114036)

[Figure 10 AUC score for 3,5,10 classes for AdaBoost with ordinal constructor, Ordinal boosting and Mode statistic over benchmark data sets 43](#_Toc63114037)

[Figure 11 Total wins over benchmark model for Random Forest model and Mode statistic using AUC score for 3,5,10 classes. 44](#_Toc63114038)

[Figure 12 Total wins over benchmark model for decision tree model and Mode statistic using AUC score for 3,5,10 classes. 44](#_Toc63114039)

[Figure 13 Total wins of AUC score of AdaBoost with ordinal decision trees and boosting using Mode statistic using over non-ordinal for 3,5 and10 classes. 45](#_Toc63114040)

# List of Tables

[Table 1 Characteristics of the Benchmark Data sets 30](#_Toc63693084)

[Table 2 Statistics functions 32](#_Toc63693085)

[Table 3 AUC score for 3 classes 33](#_Toc63693086)

[Table 4 Accuracy score for 3 classes 34](#_Toc63693087)

[Table 5 F-score score for 3 classes 34](#_Toc63693088)

[Table 6 Precision score for 3 classes 34](#_Toc63693089)

[Table 7 Recall score for 3 classes 34](#_Toc63693090)

[Table 8 MSE score for 3 classes 35](#_Toc63693091)

[Table 9 Kendall score for 3 classes 35](#_Toc63693092)

[Table 10 AdaBoost boosting effect for 3 classes 35](#_Toc63693093)

[Table 11 AUC score for 5 classes 36](#_Toc63693094)

[Table 12 Accuracy score for 5 classes 36](#_Toc63693095)

[Table 13 F-score score for 5 classes 36](#_Toc63693096)

[Table 14 Precision score for 5 classes 37](#_Toc63693097)

[Table 15 Recall score for 5 classes 37](#_Toc63693098)

[Table 16 MSE score for 5 classes 37](#_Toc63693099)

[Table 17 Kendall score for 5 classes 37](#_Toc63693100)

[Table 18 AdaBoost boosting effect for 5 classes 38](#_Toc63693101)

[Table 19 AUC score for 10 classes 38](#_Toc63693102)

[Table 20 Accuracy score for 10 classes 38](#_Toc63693103)

[Table 21 F-score score for 10 classes 39](#_Toc63693104)

[Table 22 Precision score for 10 classes 39](#_Toc63693105)

[Table 23 Recall score for 10 classes 39](#_Toc63693106)

[Table 24 MSE score for 10 classes 39](#_Toc63693107)

[Table 25 Kendall score for 10 classes 39](#_Toc63693108)

[Table 26 AdaBoost boosting effect for 10 classes 40](#_Toc63693109)

[Table 27 Recall results of Singer et al., (2020) and proposed ordinal models 46](#_Toc63693110)

[Table 28 Precision results of Singer et al., (2020) and proposed ordinal models 46](#_Toc63693111)

[Table 29 F-score results of Singer et al., (2020) and proposed ordinal models 46](#_Toc63693112)

[Table 30 MSE results of Singer et al., (2020) and proposed ordinal models 47](#_Toc63693113)

[Table 31 Kendall results of Singer et al., (2020) and proposed ordinal models 47](#_Toc63693114)

[Table 32 F-score results of Singer et al., (2020) and proposed ordinal models 47](#_Toc63693115)

[Table 33 Best Accuracy results for 5 class problems 48](#_Toc63693116)

[Table 34 Best models Accuracy results for 5 class problems 48](#_Toc63693117)

[Table 35 Best Accuracy results for 10 class problems 49](#_Toc63693118)

[Table 36 Best models Accuracy results for 10 class problems 49](#_Toc63693119)

[Table 37 Best MAE results for 5 class problems 49](#_Toc63693120)

[Table 38 Best models MAE results for 5 class problems 50](#_Toc63693121)

[Table 39 Best MAE results for 10 class problems 50](#_Toc63693122)

[Table 40 Best models MAE results for 10 class problems 50](#_Toc63693123)

[Table 41 AUC score for top proposed ordinal models compared to state-of-the-art models. 51](#_Toc63693124)

[Table 42 AUC score comparison for ensemble models 52](#_Toc63693125)

[Table 43 AUC score for ensemble models and their constructors 52](#_Toc63693126)

[Table 44 MSE score for ensemble models and their constructors 53](#_Toc63693127)

# Introduction

In ordinal classification problem, the class values exhibit a natural order. In many research studies, those problems are often solved as multi-class classification problems, while discarding the ordering form of the class which when taken into account, can improve the performance of the classifiers. Ordinal problems commonly address real-world applications such as portfolio investment by expected return performance or the classification of the severity of disease, in which the magnitude of the classification error could have critical consequences (Gaudette and Japkowicz, 2009; Cardoso and Sousa, 2011; Destercke and Yang, 2014; Gutierrez et al., 2015; Verbeke et al., 2017). Most of these techniques assume monotonicity between the explaining attributes and target (e.g., Ben-David et al., 1989; Ben-David, 1995; Christophe and Petturiti, 2013; Verbeke et al., 2017; Zhu et al., 2017). Several previous research studies have shown that ordinal classifiers yield poor classification accuracy when applied to data sets with high levels of non-monotonic noise, and that non ordinal model performances are as good as their ordinal counterparties when applied on ordinal data sets (Ben-David et al., 2009). In fact, dependencies between explanatory attributes and target class that are non-monotonic are quite common in many ordinal problems. An example for such is on medical application of congestive heart failure identification, where the ordinal attribute “blood pressure” may have a non-monotonic effect on the level of congestive heart failure (e.g., extreme blood pressure values, either high or low, may lead to a high level of congestive heart failure, while under “regular” blood pressure, the level of congestive heart failure may be low).

In a recent research (Singer et al. 2020), an ordinal decision tree based on weighted information-gain measure (WIGR) was proposed and found to be effective for classification problems in which the class variable exhibits some form of ordinal ordering, and where dependencies between the attributes and the class value can be non-monotonic. (Singer and Cohen, 2020) used the WIGR for error calculation from the maximum or minimum class and called the new measure Objective Based Information Gain (OBIG). In this research: (1) The OBIG was extended to a more generic measure in which the potential classification error was measured relative to any possible predefined class or statistic; (2) Ordinal decision tree based algorithms were developed (such as ordinal Random Forest and ordinal AdaBoost) based on the extended proposed measure, which have the precise characteristics required to address the aforementioned shortcomings of existing approaches and; (3) In order to benefit maximally from the various decision tree based algorithms, an ensemble approach was proposed which combine together the proposed ordinal algorithms with conventional algorithms to leverage the strengths of each type of classifier (Alpaydin, 2020). The proposed algorithms are naturally tuned to identify a large number of data patterns with complex dependence structures by taking into account the consequence of the magnitude of potential classification errors relative to predefined objective class or statistic.

This research contributes in a variety of ways, as listed below:

* **Direct learning of ordinal decision tree-based algorithms towards predefined target:** By generalizing the WIGR to any objective class or statistic, the splitting mechanism in decision tree construction towards identification of a desired objective can be directed.
* **Ordinal Boosting:** The proposed extended measure was adjusted and introduced into boosting algorithms for reducing bias in ordinal classification learning.
* **Leveraging strengths from ordinal and non-ordinal classifiers:** In order to gain maximum benefit from the various types of ordinal and non-ordinal algorithms, use of an ensemble approach was proposed, which combines together algorithms to leverage the strengths of each type of classifier and yield better results.
* **Practical results:** Implementation of the proposed ordinal algorithms to prediction of severity of pandemic spreading was illustrated in (Singer and Marudi, 2020), which showed to yield better results compared to other conventional algorithms.

The rest of this thesis is organized as follows. In Section 2, the related works that lead to this research are presented. In Section 3, the fundamental for this research and my contribution is explained. Section 4 describes an extension of the ordinal approach to other ordinal decision trees-based algorithms and suggests an ensemble approach combining both ordinal and non-ordinal algorithms. In Section 5, numerical experiments are designed for benchmarking the proposed algorithms with other ordinal and non-ordinal algorithms. Section 6 presents the results, and finally Section 7 concludes the research.

# Background and Related Work

Classification is one of the most common tasks in machine learning. It is used to identify to which of a set of classes a new observation belongs, based on the values of the explanatory or input variables. In ordinal classification problems, the class attribute exhibits some form of ordering, which takes into account the ranking relationship among classes. Ordinal problems commonly address real-world application problems in which the magnitude of the classification error reflects the consequences of such errors. In recent research studies, the ordinal nature of the problem has been implemented into the classifier model to solve complex problems, such as performance prediction of queuing systems (Senderovich *et al*., 2015), prediction of load level in emergency services (Senderovich *et al*., 2015; Mouroo *et al*., 2017; Sanit-in and Saikaew, 2019), or the detection of the level of congestive heart failure (Masetic and Subasi, 2016). Following Gutierrez et al., (2016), ordinal classification taxonomy currently consists of 4 approaches: (1) Naïve approaches, (2) Ordinal binary decomposition approaches; (3) Threshold models; and (4) Other advanced methods.

The naïve approaches ignore the ordering assumption of the classes, and the resulted model obtained from standard algorithms such as, regression, nominal classification and cost-sensitive classification methods. When using regression algorithms for ordinal classification, a regressor is being trained to predict a numerical value, which by rounding its prediction to the nearest ordinal value, determines the prediction of the ordinal class. Nominal classification ignores the order between the classes, which means that the misclassification error’s consequences are not taken into consideration in the model. Cost-sensitive classification is a more advanced method for ordinal classification, and is based on a cost matrix which reflects the misclassification costs between real and predicted classes in the ordinal scale. Cost sensitive methods obtain better error measures such as MAE and MSE, and usually improve conventional classification measures such as accuracy and AUC. The main drawback of this method is that when the cost-matrix is not known and pre-defined, there are several possibilities for cost matrices that reflect the ordinal scale of the problem. Furthermore, usually cost-sensitive approaches applied for different models are usually replaced by conventional techniques for construction of models, and are not adjusted in order to preserve their advantages (Ling et al., 2004).

The second approach for ordinal classification is ordinal binary decomposition. This group includes all those methods which are based on decomposing the ordinal target variable into several binary ones, using two different techniques. There are two common existing formulations of nominal decomposition. The first one is the ‘OneVsAll’ technique, which predicts in each subproblem whether a sample belongs to a certain class or not. The second is the ‘OneVsOne’ which divides the classes into subproblems for each pair of classes. There are ordinal decomposition formulations which are a variant of these two previous decompositions for nominal classification. A variant of the ‘OneVsOne’ decomposition for ordinal classification, called the ‘OneVsNext’ decomposition approach, takes into consideration the order of the classes. Variants to the ‘OneVsAll’ decomposition for ordinal classification, are called the ‘OneVsFollowers’ and the ‘OneVsPrevious’. The main drawbacks of these approaches are the complexity of the calculation and ability to obtain explainable reasons based on the features for the classification decisions.

The third approach for ordinal classification is threshold models. This approach assumes that the ordinal order can be learned from a continuous variable. The methodology of this approach defines a function that tries to map all input space into a one-dimensional continuous variable, which is a latent variable. Then, *N*-1 thresholds are assigned, where *N* is the number of ordinal classes to predict. Each interval in the range of represents a class and the model tries to learn these thresholds. This approach is different from the naïve approaches, since it does not assume in advance the distance between classes, however it is estimated during the learning process. Compared to decomposition approaches, this is a single model approach based on a single mapping function.

The fourth approach includes all methods that did not fall into the definition of any of those approaches aforementioned. An example for such approaches are decision trees-based approaches for ordinal classification problems. Decision trees are used commonly for classification tasks, while the most common decision trees algorithms are ID3, C4.5 and CRAT (Sathyadevan *et al*., 2015; Fernández-Delgado *et al*., 2014). It is known that those algorithms are interpretable and efficient. One common approach to build trees suggests using the Shannon’s entropy (Shannon, 1948) to select the splits of nodes in trees. When looking at Shannon’s entropy in the context of classification problems, it can be seen that it is often used to measure the quantity of information that is provided about the class variable by an explaining attribute (classifying attribute). It is important to mention that Shannon’s entropy, however, does not take into account the ordinal behavior of the class variables, since the entropy depends only on the distribution of possible outcomes, rather than the outcome values or their associated effects. A possible insight, for example, could be that a distribution of four symbols in a class variable that represents an expected ROI, and given by the vector (30%, 10%, 20%, 40%), will return the same entropy measure as the distribution (10%, 30%, 40%, 20%), even though the ordinality of the two vectors (and therefore their potential effects on the expected return and the associated risks) might be significantly different.

Entropy-based algorithms were extended by some research studies to consider the ordinal behavior of the class as well. However, most of them assumed that the class values follow up monotonicity constraint according to the classifying attributes (Marsala and Petturiti, 2013; Ben-David, 1995; Ben-David *et al*., 1989; Zhu*et al*., 2017; Verbeke *et al*.*,* 2017). Thus, these approaches may yield poor results in problems in which the attributes affect the class variable in non-monotonically behavior in some ranges. A non-monotonicity index was introduced by Ben-David (1995), which is defined as the ratio between the actual number of non-monotonic branch-pairs of a decision tree and the maximum number of pairs that are non-monotonic, with respect to each other in the same tree. Potharst and Bioch (2000) suggested an algorithm for repairing non-monotonic decision trees for multi-attribute classiﬁcation problems with several linearly ordered classes, and additional researchers proposed to deal with the monotonicity of data using algorithms of tree-induction (Cao-Van and Baets, 2003; Potharst and Feelders, 2002). Shannon’s entropy was generalized by Hu et al. (2010) to address fuzzy ordinal classification and crisp ordinal classification, and proposed to evaluate the degree of monotonicity between attributes and the decision in the context of ordinal classification using indices. In a later paper, Hu *et al*. (2012) designed a decision-tree algorithm based on *rank mutual information* (REMT), a new measure of attribute quality to build a monotonically consistent decision tree when the training samples are monotonically consistent. “Fusing monotonic decision trees” (FREMT), an algorithm which combines decision trees with an ensemble-learning technique, was proposed by Qian *et al.* (2015), which obtained an improved classification performance. Furthermore, some recent research studies tried to handle ordinal classification problems using ensemble techniques based on decision trees models. Ensemble methods attempt to overcome the bias or variance effects of individual classifiers by combining several of them together (Zhou, 2009), and thus achieving better performance, as shown in different research studies (Kittler et al., 1998; Bishop, 1995). The most widely used ensemble methods are categorized into four techniques: bagging, boosting, stacking, and voting (Yıldırım et al., 2019, Liang et al., 2018). Decision tree-based algorithms which implement the bagging or boosting idea to reduce bias or variance in supervised learning are proven in many cases to outscore the classical decision tree model (Ali J et el 2012, Alpaydin, 2014). Hornung (2020) proposed a model called “Ordinal Forest”, which combines the advantages of the ensemble models with the thresholds approach. Each decision tree is trained to solve a regression problem and find the optimal thresholds on subsets of features and samples. The prediction is done by the voting approach. Another approach, named Oadaboost (Costa and Cardoso, 2015), is a method that designed the AdaBoost algorithm for ordinal classification problems. The proposed method builds Binary K-1 classifiers in parallel in each iteration using the decomposition approach. The final prediction is done by aggregating all K-1 strong learners that were trained on different variants of the data sets. Both approaches, as well as most of the recent research studies, presented different combinations, usually of the first three approaches mentioned above.

Although there is a great amount of research on different ordinal classification, there is no clear conclusion on whether these ordinal classifiers perform better than non-ordinal classifiers. As a matter of fact, it was shown by Ben David (2009) that the ordinal classifiers were not statistically distinguishable from their non-ordinal counterparts, mainly since the monotonicity assumption led to high levels of non-monotonic noise, which resulted in a poor classification accuracy of the ordinal algorithms.

In this research, a general approach is presented for tree-based algorithms that utilizes the information gain approach while taking into consideration the consequences from possible errors compared to a specific predefined objective. This approach has several different aspects compared to previous approaches; (1) It enables to construct trees models based on the objective defined by decision makers, i.e., two different objectives will yield different models; (2) It utilizes the characterization of entropy measure to construct the decision tree model. and not only the misclassification costs that usually are not known in advance; (3) It does not assume monotonic behavior between the explainable features and target variable; (4) It enables flexibility to be applied for any objective and for any type of decision tree-based models; (5) The running time is equivalent to their non-ordinal counterpart models, and appropriate for any multi-class ordinal problems.

# Ordinal decision tree based algorithms

In this section the Objective Based Information Gain (OBIG) and Objective Based Entropy (OBE) methods will be extended, proposed by Singer et. al. 2020 and Singer and Cohen (2020), for an individual decision tree algorithm, which is the fundamental for this research. Specifically, the measures are generalized by introducing a general objective function defined by the classes and their distribution, which replace the specific objectives proposed in these research studies. Next, ordinal Random Forest and Ordinal AdaBoost algorithms are developed, using the general OBIG measure in the decision tree construction and boosting mechanisms, and an ensemble technique based on the proposed ordinal algorithms and non-ordinal conventional algorithms is finally suggested.

## Objective Based Information Gain (OBIG) measure

The objective based information gain (OBIG) was proposed in Singer and Cohen, 2020. This measure is based on objective based entropy (OBE), which takes into consideration the ordinal nature of the data and the magnitude of the potential classification error based on the predicted value. Assume that it is necessary to evaluate a predicted value from a data set , where denotes a sample *m* in the data set, defined by a vector of values for the *K* features, { and denotes the value of the class of a sample *m* which can get possible ordered values denoted by the random variable where denote the lowest ranking and the highest ranking. Values are defined for the different levels of as as an increasing function, such that . The values of the classes are defined such that the magnitude of potential classification errors should be considered. The OBE measure allocates weights to diﬀerent classes according to their values and dispersions with respect to the value of selected objective, over the class sub data set ,and values of classes, where represents the statistic that defines the selected class. Singer et al. (2020) proposed a measure in which the deviation of the values of the classes were calculated relative to the value of the class mode, , the most probable class in the sub data set in the node. Singer and Cohen (2020) defined two more objectives, the maximum value of the classes in the sub data set in the node and the value of the minimum value of the classes in the sub data set in the node . In this research study, the *OBE* measure will be generalized by replacing the value of selected class , with a statistical function over the data set *.* Compared to , the statistical function should not necessarily be related to a selected class, for example, it can present an expected value of the class data set, i.e.,

where is the probability that records that *m* belongs to class . Furthermore, the statistical function can be related to the entire data set or to a sub data set in a node , i.e., the EV can be calculated based on the class distribution in each node in the decision tree or can be fixed based on the entire data set. Thus, in this research, the performance will be evaluated of the proposed algorithms based on the expected values in equation (1), calculated from the entire data set (called EV\_fix), and for the data set in each node (called EV). Thus, the *OBE* measure can be generally expressed as:

Thus, is the probability that records that *m* belongs to class , represents the absolute deviation of the value of the *i*-th class, , from the value of the statistic value (i.e., the objective), , divided by the sum of all absolute difference values over all possible classes in the data set. This measure implies that an attribute with a smaller distribution around the statistical value obtains a smaller OBE value, which represents a lower risk. The factor () is a normalization factor that smooths the weights’ distribution over the different classes. The objective-based entropy measure was used for calculation of an objective-based information gain measure, for selecting branching attributes in data set in decision tree models (see Singer et al., 2020 for C4.5 model and Singer and Cohen, 2020 for CART). If the records in are partitioned over the feature , having distinct values, then the objective-based information gained is,

(3)

Where the second expression on the right of the equation is the objective-based entropy of a possible partitioning on the attribute . The value represents the weight of the *r*-th partition, and represents the objective-based entropy on the class variable , which is a subset of , in the r-th partition , given a statistical function Similarly, to the conventional information gain measure, the objective-based information gain is overly sensitive to the number of values , and thus should be normalized when a big variance exists between the number of values of different features for some algorithms (e.g., Singer and Golan, 2019). In this research, since decision tree-based algorithms are built in which the decision tree will be CART type, which branches via binary splitting at each node of the tree, for each feature, the branching features in the decision tree model based on OBIG measure without normalization will be selected. Note that the OBIG measure appropriate for handling a complex non-monotonic dependency between explanatory and target attributes may exist in many ordinal problems, as explained in Chapters 1 and 2.

## Implementing of OBIG measure in Ordinal decision tree

To construct an ordinal decision tree, the entropy and information gain formulas with their ordinal adaptation OBE and OBIG have to be replaced, and a statistic function that is suitable for the problem has to be chosen. The simplicity of this adaption allows implementation in any kind of tree-based algorithm. Figure 1 presents the pseudo code for construction of the decision trees algorithm phase and classification phase.

|  |
| --- |
| Pseudo code for Decision-Tree algorithm construction and classification |

**Preconditions:** - training data set; - set of features; - statistical function; - set of classes; - set of values for classes; .

**Decision-Tree training phase**

1 **Procedure** BuildTree

2 At each node:

3 // where is a class sub set at node

4 **return the Tree**

5**Procedure**

6

7

8 **for**  **do**

9

10 **if**

11

12

13 **end for**

14 **return**

|  |
| --- |
|  |

Figure : The meta-code of the proposed ordinal decision tree classifier based on the OBIG measure.

# Ensemble based ordinal algorithms

In this research using the objective based information gain into ensemble methods to illustrate the use of ordinal classification in ensemble algorithms is suggested. Specifically, in Section 4.1, an ordinal RF algorithm is proposed to present the use of ordinal classification in an ensemble approach based on the bagging technique. In Section 4.2, an ordinal AdaBoost algorithm is proposed to present the use of ordinal classification in an ensemble approach based on the boosting technique. In Section 4.3, an ordinal majority voting approach is proposed, based on ordinal and non-ordinal algorithms to leverage the strengths of each type of algorithm, aiming to achieve better classification results.

## Ordinal Random Forest algorithm

The random-forest algorithm is a bagging method in which random sub-samples are created from the data set with replacement to train decision trees on each sample. Since in each sub-sample the gain of each feature in the context of the target variable may be similar, and decision trees choose in each node which variable to split on, using a greedy algorithm, even with the bagging mechanism, the different decision trees may have a lot of structural similarity. Thus, in order to ensure that the trees are less correlated, in addition to choosing the samples randomly, the algorithm samples over features in each node and uses only a random subset of them to choose which variable to split on, which reduces the similarities between different decision trees. The random forest can significantly reduce the variance, and thus it is more suitable when combining individual learners that suffer from large variance. Several studies have shown that random forests yield accurate and robust classification results (Sathyadevan and Nair, 2015; Belgiu and Drăguţ, 2016; Masetic and Subasi, 2016). Figure 2 presents the pseudo code for construction of the Random Forest algorithm phase and classification phase. In the construction phase, the Random Forest is built from *L* decision trees {, each trained on *l*-th bootstrap sample of the data set, . In each node in the tree, a subset of features was randomly selected, while the attribute with the highest information-gain ratio is selected as the splitting attribute. In the classification phase, for each new instance , given to the decision trees classifiers as an input, the output is returned by each model and the class with the highest number of votes is chosen as follows:

(4)

|  |
| --- |
| Pseudo code for random-forest algorithm construction and classification |

**Preconditions:** - training data set; - set of features; - set of decision trees define Random Forest; - statistical function; - set of classes; - set of values for classes; ; - unlabeled instance;

**Random Forest training phase**

1 **Procedure** BuildRandomForest

2

3 **for**  **do**

4 A bootstrap sample from

5  BuildTree

6

7 **end for**

8 **return**

9 **Procedure** BuildTree

10 At each node:

11 A bootstrap sample from

12

13 **return the Tree**

14 **Procedure**

15

16

17 **for**  **do**

18

19 **if**

20

21

22 **end for**

|  |
| --- |
| 23 **return**  **Classification phase**  24 **Procedure** Classify  25 BuildRandomForest  26  27 **return** |

Figure The meta-code of the proposed ordinal random-forest classifier based on the OBIG measure.

## Ordinal boosting algorithm

The AdaBoost algorithm is a boosting adaptive method in the sense that subsequent weak classifiers are tweaked in favor of those instances misclassified by a previous classifier, i.e., the classifiers are built sequentially and not in parallel like in the Random Forest algorithm. Each classifier built focuses on previous harder-to-classify examples by increasing their weights in the data sets. In this example, 1-level decision trees are used based on the OBIG measure as the weak classifiers (which are low complexity models). The AdaBoost can significantly reduce the bias, i.e., difference between the predicted value and true target value relative to the training data, which usually occurs due to low complexity of the model. Similar to the Random Forest, AdaBoost makes classification by applying multiple decision trees to every sample and combining the predictions made by individual trees. However, rather than taking the majority voting between the decision trees in the Random Forest, in the AdaBoost algorithm, every decision tree contributes a varying amount to the final prediction according to the incorrectly classified samples. Several studies have shown that AdaBoost yield accurate and robust results (Wang et. al., 2006; Vezhnevets and Vezhnevets, 2005; Sun et. al., 2016). Figure 3 presents the pseudo code for construction of the AdaBoost model built from 1-level decision trees based on the OBIG measure phase and classification phase. In the construction phase, the AdaBoost built *L* decision trees with a depth of 1 {, sequentially, each trying to correct its predecessor and trained on a data set, generated from the sample’s weights, as will be explained later. In the procedure BuildTree, in each node the attribute with the highest information-gain ratio is selected as the splitting attribute, using the procedure . In each iteration in the procedure Build AdaBoost, the model was built based on a training data set that will have a tendency to contain multiple copies of the samples that were misclassified by a previous decision tree. The mechanism for generating different copies for each sample is based on weights values for iteration *l*. The first step begins with *l=*1, with identical weights for the *M* samples in the data set, to build the decision tree. In each iteration, the error is calculated as follows:

(5)

Thus, is an error function which assigns value 1 when a predicted value of a sample different from the real value and assigns 0 otherwise. Note that is required, where *n* is the number of classes, such that if the error exceeds the upper value (which reflects the error of the naïve classifier, assuming equal probabilities distribution for the classes) no more decision trees will be generated. Note that the error in each sample gets the same value 1, regardless of the magnitude of the classification error and consequences, as considered in the OBE measure and decision tree building. Alternative option replacing equation (5) by considering the magnitude of the classification error can be, Suggestion – An alternative option could be by replacing equation (5) by considering the magnitude of the classification error, which could yield:

(6)

such that an error value assigned according to the difference between the values of the predicted and actual class, divided by the maximum difference between two different class values to normalize the error to the range . The error is used to calculate the decision tree weight in the final classification as follows:

(7)

where and its value increases as the classifier’s error decreases, while when , which means the weight of a classifier tends to zero as its error tends to the naïve classifier’s error. is used for updating the sample weights, such that misclassified samples will get higher weights as follows:

(8)

and is the normalization constant such that the new sample weights will sum up to 1. Note that when the error of the classifier tends to the naïve classifier error or equal to zero (i.e., all samples classified correctly), then the weights of the samples will not change compared to the previous iteration up to a constant. In any other cases, decreases for samples that were correctly classified by . Similarly to equation (5), the error of a sample can be replaced by equation (6) to consider the magnitude of the classification error of each sample. In the classification phase, for each new instance , given to the decision trees models as an input, the output is returned by each model and contributes to the final prediction as follows:

(9)

|  |
| --- |
| Pseudo code for AdaBoost algorithm construction and classification |

**Preconditions:** - training data set; - set of features; - set of decision trees of the AdaBoost; - set of weights of decision trees; - statistical function; - set of *n* classes; - set of values for classes; ; - unlabeled instance; - initial weights set to

**AdaBoost training phase**

1 **Procedure** BuildAdaBoost

2 ,

3 **for**  **do**

4 An updated generated data set, drawn from the probabilities set

5 BuildTree

6

7

8 if then set L=l-1 and abort the loop

8

9

10

7 **end for**

8 **return**

9 **Procedure** BuildTree

10 At each node:

12

13 **return the Tree**

14 **Procedure**

15

16

17 **for**  **do**

18

19 **if**

20

21

22 **end for**

|  |
| --- |
| 23 **return**  **Classification phase**  24 **Procedure** Classify  25 BuildAdaBoost  26  27 **return** |

Figure The meta-code of the proposed ordinal AdaBoost classifier based on the OBIG measure

## Majority voting approach

In order to benefit maximally from the various decision trees algorithms based on a variety of objective-based entropies, a simple ensemble approach is proposed to combine them together, which in theory, should leverage the strengths of each type of classifier (Alpaydin, 2020, Singer et al, 2020). Despite the fact that ensemble methods for nominal classification have received considerable attention in the literature, and have been chosen in preference to single, individual learning algorithms for many classification tasks, the use of ordinal classification in ensemble algorithms has rarely been discussed (Yıldırım et al., 2019). Figure 4 presents a pseudo code for implementation of the majority voting approach based on *L* decision tree-based models . The procedure BuildModel, call to one of the decision-trees based models’ construction procedure such as the ordinal decision tree, AdaBoost or Random Forest in Figures 1-3. The trees of each model is constructed using object-based information gain measures calculated with procedure OBIG according to statistic , where define the entire set of statistics. Each one of the ordinal decision-trees based algorithms is constructed based on the entire training data set . After building the models, for each new sample given to the classifiers as an input, the output is returned by each model and the class with the highest number of votes is chosen following equation (4).

|  |
| --- |
| Pseudo code for Majority Voting approach of a sample *X* |

**Preconditions**: – training data set*;*- set of features; *G* – set of tree-based models; – set of statistics ; - set of classes; - set of values for classes; - unlabeled instance;

**Ensemble approach training phase**

1 **Procedure** BuildDTModels

2 ,

3 for do

4 BuildModel

5

6 end for

7 **return**

|  |
| --- |
| **Classification phase**  8 **Procedure** Classify  9 BuildDTModels  10  11 **return** |

Figure Pseudo code for majority voting approach based on *L* decision tree-based algorithms

# Experiment

In this Section, evaluation of the ordinal decision-trees based algorithms for ordinal classification problems, with class values that exhibit a natural order, is pursued. For this purpose, 10 ordinal data sets were used to benchmark the performance of the proposed algorithms with other well-known state-of-the-art algorithms.

The subgoals of this section are:

* Compare the proposed ordinal decision tree-based algorithms with their counterparts using 7 common performance measures, including hypothesis tests that prove statistical significance difference.
* Compare the proposed ordinal decision-tree-based algorithms to the well-known-state-of-the-art algorithms.
* Evaluate the contribution of combining ordinal and non-ordinal algorithms to leverage the strengths of each type of classifier.

## Data description

The experiment was conducted over ten publicly known data sets, which were previously published and evaluated by Gutierrez et al. (2016) and Singer et al. (2020).

Gutierrez et al. (2016) reviewed the different approaches to solve ordinal classification problems and compared 16 ordinal models, belonging to different ordinal methods as explained in Chapter 2. The comparison was conducted by 2 known metrics: accuracy and mean absolute error (MAE). Singer et al. (2020) compared ordinal C4.5 with the mode statistic and a normalization factor equal to 1 (α=1) C4.5, with conventional decision tree to C4.5 using the 5 common used metrices: F-score, recall, precision, MSE and Kendall correlation to evaluate their results.

The 10 mentioned data sets had continuous predicted values on an interval scale which were discretized by Gutierrez et al. (2016) into 10 ranked bins with equal frequency of samples, in order to apply and evaluate ordinal classification algorithms. To evaluate the effect of the number of classes on the performance of the proposed ordinal algorithms, following Singer et al., (2020), bins were merged to create 3 and 5 class problems. In the 3-class problems, classes 1-3 were labelled as ‘Low’, classes 4-7 as ‘Medium’ and classes 8-10 as ‘High’ and in the 5-class problems, classes 1-2 were labelled as ‘Very Low’, classes 3-4 as ‘Low’, classes 5-6 as ‘Medium’, classes 7-8 as ‘High’ and classes 9-10 as ‘Very High’.

Table 1 presents for each data set: the number of samples (rows), number of features and number of samples per class for the three options for a different number of classes (3,5 and 10 classes). Those data sets have different characteristics which will enable evaluation of the robustness of the proposed algorithms to yield good results.

Table Characteristics of the Benchmark Data sets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Set name | Rows | Features | ~Rows per class | | |
| 3 class | 5 class | 10 class |
| abalone | 4177 | 11 | 1392 | 835 | 418 |
| bank1 | 8192 | 9 | 2731 | 1638 | 819 |
| bank2 | 8192 | 33 | 2731 | 1638 | 819 |
| calhousing | 20640 | 9 | 6880 | 4128 | 2064 |
| census1 | 22783 | 9 | 7594 | 4557 | 2278 |
| census2 | 22783 | 17 | 7594 | 4557 | 2278 |
| computer1 | 8192 | 13 | 2731 | 1638 | 819 |
| computer2 | 8192 | 22 | 2731 | 1638 | 819 |
| housing | 506 | 14 | 169 | 101 | 51 |
| machine | 198 | 7 | 66 | 39 | 20 |
| pyrim | 74 | 27 | 25 | 15 | 7 |
| stock | 950 | 10 | 317 | 190 | 95 |

## Performance criteria and evaluation method

For the purpose of benchmarking, the performance of the ordinal and non-ordinal classifiers was computed using a total of seven performance measures for multi-class classiﬁcation as follows: F-score, Accuracy, Recall, Precision and Area Under the Curve (AUC) (Sokolova and Lapalme, 2009). Additionally, the mean squared error (MSE) was used and Kendall’s correlation coeﬃcient,, which are acceptable performance measures for ordinal classification problems (Cardoso and Costa, 2007; Singer and Cohen, 2020; Singer and Marudi, 2020).

In order to calculate the mean squared error (MSE), the following equation was used,

where is the number of samples, is the real class of sample , is the estimated class, returned by the model of sample , and corresponds to a number assigned to each class, in this case, . A lower MSE score represents a smaller classification error. Another interesting performance measure is Kendall’s correlation coeﬃcient, which is a measure of concordance or the ordinal association between two measured quantities. It is defined as:

(11)

where refers to the number of concordant pairs in the classification, i.e., pairs in which the relative ordering of the “real” classes and is the same as the relative ordering of the classified classes and , and refers to the number of discordant pairs in the classification, i.e., pairs in which the relative ordering of the “real” classes is opposite to the relative ordering of the classified classes. The parameter refers to the number of samples that are tied on the true class and refers to the number of samples that are tied on the estimated class. Pairs that hold the same real and estimated classes are ignored. ranges from -1 to 1. The higher the performance measure, the better the classification performance.

To evaluate the best classification model, a 5-fold cross validation technique was used, while keeping the same distribution of classes in each fold, as conducted in Gutierrez *et al*. (2016). The classification measures are evaluated and averaged over the 5 folds, each time leaving out one of the sub-samples and using it as a test case. For each OBIG tree-based model, the normalization factor () was optimized by running a grid search over 10 possible values between 0 and 5. To choose the best model, the average AUC value over 5 folds based on the training samples was used. The minimum samples in each node were defined to be 10 for all decision tree-based models. The t-test was performed to determine the statistical significance of the performance difference between algorithms.

## Explanation of experimental stages and models used

In this section, the different stages of the experiment, and the models used will be explained. In the first stage, the proposed ordinal decision tree-based algorithms with their non-ordinal counterparts were benchmarked. The ordinal models used were decision tree, AdaBoost and Random Forrest, each with five different statistic functions (as explained in Section 3.1) and presented in Table 2.

Table Statistics functions

|  |  |  |
| --- | --- | --- |
| **Notation** |  | Explanation |
|  |  | Maximum value among all classes |
|  |  | Value of the most probable class in the training data set |
|  |  | Minimum value among all classes |
|  |  | Expected value of the class’s distribution in sub data set in a node |
|  |  | Expected value of the class’s distribution in the training data set |

In this section the notations Max, Mode, Min, EV and EV\_fix will be used to indicate the statistical function used in the OBIG to construct the decision trees. In addition, to testing the ordinal boosting approach, the conventional AdaBoost was compared to the proposed ordinal AdaBoost with ordinal decision trees (both with and without and ordinal boosting process).

In the second stage, the best performed ordinal decision trees based algorithms with ordinal and non-ordinal algorithms were benchmarked. Practically, the performance with the commonly used ordinal algorithm, proposed in Frank and Hall (2001), was compared to the best results achieved from 16 ordinal algorithms in Gutierrez et al. (2016), and state-of-the-art non-ordinal algorithms, such as logistic regressions, XGBoost and GBM.

In the third stage, 2 ensemble models were combined (one from only ordinal algorithms and one from ordinal and non-ordinal algorithms), in order to evaluate the contribution of combining different types of algorithms.

# Results and Discussion

## Results of ordinal decision tree-based algorithms vs. non-ordinal counterparts.

The performance measures of the 3-classes data sets for each one of the proposed ordinal decision trees-based algorithms were compared to its non-ordinal counterparts, as presented in Tables 3-9. The rows represent the statistical function used in the OBIG criterion to construct the ordinal decision trees within the algorithms. Note that the third column represents comparison between the conventional AdaBoost and AdaBoost with ordinal decision trees, while the last column represents comparison between the conventional AdaBoost and AdaBoost with ordinal decision trees and ordinal boosting. Each cell contains a tuple, with the first value representing the number of wins: the number of data sets in which the ordinal model outperformed its non-ordinal counterpart, and second value, represents the number of data sets that the wins were statistically significant with (accepted from a paired ).

Table AUC score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (9,7) | (10,9) | (5,3) | (8,6) |
| max | (10,7) | (10,9) | (5,3) | (8,5) |
| min | (10,7) | (10,9) | (7,5) | (8,5) |
| EV | (9,7) | (10,8) | (9,6) | (9,3) |
| EV\_fix | (10,7) | (10,9) | (6,3) | (8,4) |

Table 3 presents the results of the AUC score. The ordinal decision trees achieved a total of 96% success rate, outperforming 48 out of 50 of the cases, with 70% of the cases being statistically significant. The random forest yielded 100% success rate with 88% cases that are statistically significant. The AsaBoost with ordinal decision trees and boosting yielded better results than the AdaBoost, with ordinal decision trees having only a 82% success rate and 46% cases that are statistically significant.

Table Accuracy score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (8,5) | (10,9) | (4,3) | (7,6) |
| max | (9,7) | (10,9) | (2,1) | (5,4) |
| min | (9,7) | (10,10) | (5,1) | (7,4) |
| EV | (9,7) | (10,10) | (9,7) | (7,3) |
| EV\_fix | (9,7) | (10,10) | (4,0) | (5,3) |

Table F-score score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (8,6) | (10,10) | (5,3) | (7,6) |
| max | (9,7) | (10,9) | (2,1) | (5,4) |
| min | (9,7) | (10,10) | (5,0) | (7,4) |
| EV | (8,7) | (10,10) | (9,7) | (7,3) |
| EV\_fix | (9,7) | (10,10) | (5,3) | (5,3) |

Table Precision score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (9,5) | (10,9) | (3,2) | (7,6) |
| max | (9,7) | (10,9) | (2,1) | (5,3) |
| min | (8,7) | (10,8) | (4,0) | (6,3) |
| EV | (6,4) | (10,8) | (9,5) | (8,3) |
| EV\_fix | (9,7) | (10,8) | (4,0) | (4,3) |

Table Recall score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (8,5) | (10,8) | (4,2) | (8,5) |
| max | (9,7) | (10,8) | (3,2) | (6,4) |
| min | (9,6) | (10,9) | (6,1) | (7,4) |
| EV | (9,6) | (10,9) | (8,6) | (7,3) |
| EV\_fix | (9,7) | (10,8) | (4,0) | (5,3) |

In Tables 4 – 7, the results for the 3 classes experiment over 4 known performance measures can be seen: Accuracy, F-score, precision and recall. The results maintained the same observed trend, as can be seen in Table 1.

Table MSE score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (8,6) | (10,10) | (5,3) | (7,5) |
| max | (9,7) | (10,9) | (2,1) | (6,4) |
| min | (9,7) | (10,10) | (5,0) | (7,4) |
| EV | (8,7) | (10,10) | (9,7) | (7,4) |
| EV\_fix | (9,7) | (10,10) | (3,0) | (6,3) |

Table Kendall score for 3 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,9) | (4,3) | (7,6) |
| max | (8,8) | (10,10) | (4,2) | (6,5) |
| min | (8,6) | (10,9) | (3,0) | (7,4) |
| EV | (8,7) | (9,9) | (8,7) | (7,4) |
| EV\_fix | (7,7) | (10,9) | (4,0) | (6,3) |

Tables 8 and 9 present performance measures commonly used for ordinal classification problems emphasizing the magnitude of errors. Both tables show the same trend as seen in previous results, while during this experiment, the proposed ordinal decision tree models were tested and compared to their non-ordinal counterpart models. It was seen that the ordinal random forest and decision trees achieved significant better results, regardless of the statistic function used. The proposed AdaBoost with both ordinal decision trees and ordinal boosting achieved better results compared to the model with only the ordinal decision trees. Interestingly enough, it was observed that models that used the EV function showed relatively good and stable results with a minimum of 7 wins, regardless of the model used.

Table AdaBoost boosting effect for 3 classes

|  |  |
| --- | --- |
| **Metric** | **Win / Significant rate** |
| Roc | (7,4) |
| Accuracy | (7,4) |
| F-score | (7,4) |
| Precision | (7,3) |
| Recall | (7,4) |
| MSE | (5,4) |
| Kendal | (6,5) |

To test the effect of the ordinal boosting approach, the AdaBoost was compared with non-ordinal decision trees and ordinal boosting with a conventional AdaBoost, as shown in Table 10. As can be seen, the results were similar to the model with both ordinal decision trees and ordinal boosting. However, it is important to mention that the ordinal AdaBoost with only ordinal decision trees with the EV statistical function yield the best results for all performance measures.

The performance measures of the 5-classes data sets, for each one of the proposed ordinal decision trees-based approach compared to its non-ordinal counterparts, are presented in Tables 11-17

Table AUC score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,9) | (4,3) | (7,6) |
| max | (8,8) | (10,10) | (4,2) | (6,5) |
| min | (8,6) | (10,9) | (3,0) | (5,5) |
| EV | (8,7) | (9,9) | (8,7) | (5,2) |
| EV\_fix | (7,7) | (10,9) | (4,0) | (7,5) |

Table Accuracy score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,9) | (4,1) | (9,6) |
| max | (10,8) | (10,9) | (2,0) | (6,5) |
| min | (10,8) | (10,9) | (3,0) | (5,5) |
| EV | (9,7) | (10,9) | (9,4) | (5,3) |
| EV\_fix | (10,9) | (10,9) | (3,0) | (6,5) |

Table F-score score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,9) | (4,1) | (7,6) |
| max | (9,8) | (10,8) | (1,0) | (6,4) |
| min | (10,8) | (10,8) | (3,0) | (6,5) |
| EV | (9,7) | (9,6) | (9,5) | (5,2) |
| EV\_fix | (10,9) | (10,8) | (3,0) | (6,6) |

Table Precision score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,10) | (4,1) | (7,4) |
| max | (9,8) | (10,9) | (2,0) | (5,3) |
| min | (10,9) | (10,9) | (3,0) | (6,3) |
| EV | (9,8) | (9,7) | (9,5) | (4,3) |
| EV\_fix | (10,9) | (10,9) | (3,0) | (6,4) |

Table Recall score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,9) | (4,1) | (8,6) |
| max | (10,8) | (10,9) | (2,0) | (6,5) |
| min | (10,8) | (10,9) | (4,0) | (5,5) |
| EV | (9,7) | (10,8) | (9,5) | (5,2) |
| EV\_fix | (10,9) | (10,9) | (3,0) | (6,5) |

Table MSE score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,10) | (7,2) | (7,6) |
| max | (10,8) | (10,9) | (3,1) | (6,5) |
| min | (9,7) | (10,9) | (5,1) | (6,5) |
| EV | (10,7) | (8,5) | (8,5) | (4,2) |
| EV\_fix | (10,9) | (10,9) | (6,0) | (6,4) |

Table Kendall score for 5 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,6) | (10,10) | (6,2) | (7,6) |
| max | (10,6) | (10,10) | (3,0) | (6,5) |
| min | (9,6) | (10,9) | (4,1) | (6,5) |
| EV | (10,7) | (9,9) | (9,5) | (6,3) |
| EV\_fix | (10,7) | (10,10) | (5,0) | (6,6) |

In this stage of the experiment, similar results to 3-class problems are seen. Ordinal random forest and ordinal decision trees showed superior results over their counterparts. Random forest achieved at least a 96% winning rate over all performance measures and at least 84% of those results were statistically significant. The ordinal decision tree models also scored at least a 96% winning rate for all performance measures except AUC, which had an 82% winning rate. Ordinal AdaBoost with only ordinal decision trees showed poor performance over all performance measures and statistical functions except the EV statistic. AdaBoost with both ordinal decision trees and ordinal boosting showed better results. As mentioned above, the EV showed a different performance, which will be elaborated later in this section. As can be seen in Table 18, the ordinal boosting was able to lift the results against the conventional boosting methods, with similar results to the AdaBoost with both ordinal decision trees and ordinal boosting.

Table AdaBoost boosting effect for 5 classes

|  |  |
| --- | --- |
| Metric | Win / Significant rate |
| Roc | (5,5) |
| Accuracy | (5,5) |
| F-score | (6,4) |
| Precision | (5,4) |
| Recall | (5,5) |
| MSE | (6,5) |
| Kendal | (6,5) |

The performance measures of the 10-classes data sets, for each one of the proposed ordinal decision trees-based model, in comparison to its non-ordinal counterpart, are compared in Tables 20-26.

Table 19 AUC score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (9,8) | (10,9) | (6,2) | (10,8) |
| max | (10,7) | (9,8) | (3,1) | (9,8) |
| min | (9,8) | (9,8) | (4,0) | (9,8) |
| EV | (8,7) | (9,8) | (2,2) | (7,5) |
| EV\_fix | (10,9) | (10,9) | (5,0) | (9,8) |

Table Accuracy score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (8,7) | (10,9) | (6,2) | (10,8) |
| max | (9,7) | (9,8) | (3,0) | (9,8) |
| min | (9,8) | (9,8) | (6,1) | (9,8) |
| EV | (8,7) | (9,8) | (3,2) | (7,5) |
| EV\_fix | (8,7) | (9,8) | (3,2) | (9,8) |

Table F-score score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,9) | (10,9) | (6,2) | (10,9) |
| max | (10,9) | (10,9) | (5,0) | (10,9) |
| min | (10,9) | (10,9) | (7,1) | (10,9) |
| EV | (10,8) | (9,6) | (3,1) | (6,4) |
| EV\_fix | (10,9) | (10,9) | (6,0) | (10,9) |

Table Precision score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,9) | (9,8) | (6,2) | (10,8) |
| max | (10,9) | (9,8) | (4,0) | (10,8) |
| min | (10,9) | (9,8) | (8,1) | (10,8) |
| EV | (10,9) | (9,7) | (4,2) | (6,4) |
| EV\_fix | (10,10) | (9,9) | (5,0) | (10,7) |

Table Recall score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (9,8) | (10,9) | (6,2) | (10,7) |
| max | (10,7) | (9,8) | (4,0) | (9,8) |
| min | (10,9) | (9,8) | (5,1) | (9,8) |
| EV | (8,7) | (9,8) | (3,2) | (6,4) |
| EV\_fix | (10,9) | (10,9) | (4,0) | (9,8) |

Table MSE score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (10,8) | (10,10) | (8,2) | (10,5) |
| max | (10,8) | (10,9) | (8,0) | (10,5) |
| min | (10,8) | (10,9) | (4,1) | (9,4) |
| EV | (10,8) | (9,7) | (3,3) | (3,2) |
| EV\_fix | (10,8) | (10,9) | (0,0) | (8,4) |

Table Kendall score for 10 classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistic | Decision Tree vs. ordinal Decision Tree | Random Forest vs. ordinal Random Forest | AdaBoost vs. AdaBoost with ordinal decision trees | AdaBoost vs. AdaBoost with ordinal decision trees and ordinal boosting |
| mode | (9,6) | (10,10) | (5,2) | (8,6) |
| max | (7,5) | (10,9) | (4,0) | (8,5) |
| min | (9,6) | (10,10) | (4,1) | (8,5) |
| EV | (8,4) | (10,9) | (5,4) | (4,2) |
| EV\_fix | (9,6) | (10,10) | (3,0) | (9,5) |

Table AdaBoost boosting effect for 10 classes

|  |  |
| --- | --- |
| Metric | Win / Significant rate |
| Roc | (10,8) |
| Accuracy | (10,8) |
| F-score | (10,9) |
| Precision | (10,8) |
| Recall | (10,8) |
| MSE | (9,4) |
| Kendal | (7,5) |

For the 10 classes experiment, it was observed that the proposed ordinal models continued the same trend as in previous experiments, and even yielded better results. In Figure 5, it can be seen that the AUC score decreases per data set when comparing the results of 3 and 5 classes and the results of 5 and 10 classes, for the non-ordinal models (graph on the left-hand side) and for the proposed ordinal models (in the graph on the right-hand side). The graphs demonstrate that for non-ordinal models, an increase in the number of classes from 5 to 10 classes resulted with a higher decrease, compared to the proposed ordinal models where the decrease in performance is more moderate.

Figure AUC relative decreasing score per data set non-ordinal models

Figure AUC relative decreasing score per data set ordinal models

To conclude it can be seen that the proposed ordinal random forest and decision tree models yield better results than the conventional non-ordinal counterparts. The data sets (pyrim, stocks, machine) in which the ordinal AdaBoost yields poor results have the lowest number of samples which led to the assumption that this is the reason the boosting process was not effective. Although the AdaBoost, both with ordinal decision trees and boosting with the EV statistical function, achieved good results.

In order to evaluate the stability of results along a different number of classes, comparison was made with the performance of the ordinal decision tree-based models using the mode statistical function.

Each one of graphs 6-9 presents the performance of one of the ordinal decision tree-based algorithm over all data sets for 3, 5 and 10 classes. The dark (black color) bars represent the original results for non-ordinal models, and the brighter (grey color) bars represent the proportional improvement presented by the ordinal model. For each data set, the first bar presents the 3-class problem, the second bar presents the 5-bars problem and the third bar presents the 10-bar problems. It can be seen that the proposed ordinal algorithms outperformed their non-ordinal counterparts and yield similar achievements, regardless of the number of classes. Graphs 10-13 present the performance of the proposed ordinal decision tree-based models over the entire data sets. It can be seen that the performance is stable regardless of the number of classes.



Figure AUC score for 3,5,10 class for Random Forest model and Mode statistic over benchmark data sets



Figure AUC score for 3,5,10 classes for Decision Tree model and Mode statistic over benchmark data sets



Figure AUC score for 3,5,10 classes for AdaBoost with ordinal constructor and Mode statistic over benchmark data sets



Figure AUC score for 3,5,10 classes for AdaBoost with ordinal constructor, Ordinal boosting and Mode statistic over benchmark data sets

Figure Total wins over benchmark model for Random Forest model and Mode statistic using AUC score for 3,5,10 classes.

Figure Total wins over benchmark model for Decision Tree model and Mode statistic using AUC score for 3,5,10 classes.

Figure Total wins of AUC score of AdaBoost with ordinal decision trees and boosting using Mode statistic using over non-ordinal for 3,5 and10 classes.

## Comparison of the proposed ordinal decision tree-based models results with previous published results

In this sub-section, a comparison will be presented between the results of the proposed ordinal models with those published in Singer et al. (2020) and Gutierrez et al. (2016).

Tables 27-31 present the results of decision tree C4.5 and ordinal decision tree C4.5 from Singer et al. (2020) for the 3-class problems, compared to the proposed ordinal decision tree-based algorithms. Each table represents a different performance measure. In Table 32, only the F-score performance measure for the 5 class problems was compared with the results. The best results were marked with grey background.

It can be seen that the proposed ordinal random forest has yielded the best results, outscoring its competitors in each performance measure in 8 of 10 data sets. The best scores in the remaining 2 data sets belonged to the other ordinal models in this research.

Table Recall results of Singer et al., (2020) and proposed ordinal models

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T-Test on the Recall Measure | Regular C4.5 | Ordinal Decision Tree C4.5 | CART | Ordinal Decision Tree CART | Ordinal Random Forest | Ordinal AdaBoost1 | Ordinal AdaBoost2 |
| abalone | 57.80% | 63.30% | 57.82% | 59.66% | 64.42% | 58.32% | 59.30% |
| bank1 | 72.70% | 80.80% | 68.62% | 80.36% | 87.54% | 83.67% | 85.63% |
| calhousing | 64.50% | 69.20% | 59.89% | 73.95% | 79.22% | 74.20% | 75.89% |
| census1 | 67.70% | 73.50% | 60.98% | 62.66% | 74.26% | 69.92% | 70.88% |
| computer1 | 76.90% | 80.10% | 70.14% | 70.13% | 83.31% | 72.75% | 81.69% |
| computer2 | 76.30% | 80.00% | 69.03% | 76.26% | 86.19% | 77.16% | 83.72% |
| housing | 74.10% | 75.80% | 77.28% | 76.49% | 79.03% | 72.22% | 75.10% |
| machine | 67.60% | 71.20% | 66.67% | 71.11% | 76.28% | 74.28% | 69.65% |
| pyrim | 35.80% | 39.50% | 52.92% | 63.75% | 47.50% | 52.08% | 58.19% |
| stock | 87.10% | 88.90% | 89.89% | 92.63% | 93.58% | 94.84% | 92.42% |

Table Precision results of Singer et al., (2020) and proposed ordinal models

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T-Test on the Precision | Regular C4.5 | Ordinal Decision Tree C4.5 | CART | Ordinal Decision Tree CART | Ordinal Random Forest | Ordinal AdaBoost1 | Ordinal AdaBoost2 |
| abalone | 58.40% | 61.30% | 57.02% | 59.72% | 64.84% | 58.37% | 59.41% |
| bank1 | 73.60% | 80.50% | 72.30% | 79.79% | 88.99% | 83.04% | 85.18% |
| calhousing | 63.90% | 67.60% | 67.02% | 72.18% | 81.45% | 74.99% | 76.17% |
| census1 | 71.40% | 72.20% | 65.28% | 65.51% | 73.87% | 70.35% | 71.17% |
| computer1 | 77.10% | 78.90% | 72.48% | 72.94% | 82.01% | 73.30% | 81.84% |
| computer2 | 76.40% | 78.60% | 73.95% | 76.45% | 86.52% | 77.43% | 83.41% |
| housing | 74.60% | 76.20% | 80.50% | 73.58% | 82.42% | 73.78% | 72.88% |
| machine | 68.40% | 72.30% | 69.00% | 74.77% | 77.90% | 76.02% | 71.63% |
| pyrim | 39.20% | 41.70% | 49.27% | 61.88% | 58.13% | 55.63% | 63.74% |
| stock | 87.70% | 87.90% | 90.44% | 92.96% | 93.83% | 95.00% | 92.77% |

Table F-score results of Singer et al., (2020) and proposed ordinal models

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T-Test on F-Score Measure | Regular C4.5 | Ordinal Decision Tree C4.5 | CART | Ordinal Decision Tree CART | Ordinal Random Forest | Ordinal AdaBoost1 | Ordinal AdaBoost2 |
| abalone | 56.50% | 61.30% | 56.77% | 58.81% | 64.50% | 58.23% | 59.28% |
| bank1 | 72.80% | 80.00% | 67.46% | 79.41% | 87.47% | 83.55% | 85.61% |
| calhousing | 63.90% | 67.30% | 59.12% | 73.26% | 79.30% | 74.09% | 76.02% |
| census1 | 70.40% | 71.50% | 61.10% | 63.67% | 73.62% | 70.21% | 71.02% |
| computer1 | 76.90% | 78.30% | 69.96% | 67.84% | 83.38% | 72.22% | 81.72% |
| computer2 | 76.10% | 78.30% | 68.39% | 75.76% | 86.25% | 76.79% | 83.73% |
| housing | 74.00% | 75.60% | 76.85% | 72.99% | 81.49% | 71.93% | 72.31% |
| machine | 68.00% | 70.10% | 64.92% | 68.92% | 75.63% | 73.58% | 71.12% |
| pyrim | 48.90% | 50.40% | 48.63% | 60.94% | 41.08% | 55.53% | 57.17% |
| stock | 87.20% | 88.00% | 89.94% | 94.95% | 96.53% | 94.96% | 92.41% |

Table MSE results of Singer et al., (2020) and proposed ordinal models

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T-Test on MSE | Regular C4.5 | Ordinal Decision Tree C4.5 | CART | Ordinal Decision Tree CART | Ordinal Random Forest | Ordinal AdaBoost1 | Ordinal AdaBoost2 |
| abalone | 0.5021 | 0.4917 | 0.0546 | 0.0579 | 0.0521 | 0.0329 | 0.5011 |
| bank1 | 0.2779 | 0.2042 | 0.0175 | 0.0315 | 0.0110 | 0.0238 | 0.1437 |
| calhousing | 0.4723 | 0.4358 | 0.0168 | 0.0344 | 0.0165 | 0.0311 | 0.2722 |
| census1 | 0.3627 | 0.3474 | 0.0193 | 0.0271 | 0.0169 | 0.0250 | 0.3279 |
| computer1 | 0.2521 | 0.2332 | 0.0225 | 0.0237 | 0.0176 | 0.0367 | 0.2123 |
| computer2 | 0.2568 | 0.2286 | 0.0191 | 0.0218 | 0.0120 | 0.0303 | 0.1691 |
| housing | 0.3012 | 0.2788 | 0.1284 | 0.1478 | 0.1078 | 0.1713 | 0.2786 |
| machine | 0.3681 | 0.3418 | 0.1207 | 0.1607 | 0.0957 | 0.1014 | 0.3027 |
| pyrim | 0.9874 | 0.9937 | 0.3664 | 0.5593 | 0.3418 | 0.3688 | 0.5781 |
| stock | 0.1329 | 0.123 | 0.0136 | 0.0243 | 0.0206 | 0.0107 | 0.0758 |

Table Kendall results of Singer et al., (2020) and proposed ordinal models

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T-Test on Kendall Tau | Regular C4.5 | Ordinal Decision Tree C4.5 | CART | Ordinal Decision Tree CART | Ordinal Random Forest | Ordinal AdaBoost1 | Ordinal AdaBoost2 |
| abalone | 0.3757 | 0.4464 | 0.5572 | 0.5356 | 0.6099 | 0.5160 | 0.5282 |
| bank1 | 0.5759 | 0.7034 | 0.8026 | 0.7608 | 0.8847 | 0.8277 | 0.8523 |
| calhousing | 0.4437 | 0.5195 | 0.7059 | 0.6643 | 0.7957 | 0.7291 | 0.7383 |
| census1 | 0.5233 | 0.5932 | 0.5929 | 0.5788 | 0.7195 | 0.6673 | 0.6770 |
| computer1 | 0.6344 | 0.6921 | 0.7353 | 0.7317 | 0.8292 | 0.7368 | 0.7867 |
| computer2 | 0.6263 | 0.6917 | 0.7668 | 0.7603 | 0.8562 | 0.7751 | 0.8264 |
| housing | 0.5910 | 0.6210 | 0.7607 | 0.6917 | 0.7803 | 0.7019 | 0.6795 |
| machine | 0.5024 | 0.5625 | 0.7645 | 0.7575 | 0.7710 | 0.7489 | 0.6850 |
| pyrim | 0.0360 | 0.1391 | 0.6911 | 0.5780 | 0.5891 | 0.5820 | 0.5625 |
| stock | 0.7920 | 0.8196 | 0.9217 | 0.9016 | 0.9346 | 0.9459 | 0.9220 |

Table F-score results of Singer et al., (2020) and proposed ordinal models

for 5 class data sets

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T-Test on F-Score Measures  5 class | Regular C4.5 | Ordinal Decision Tree C4.5 | CART | Ordinal Decision Tree CART | Ordinal Random Forest | Ordinal AdaBoost1 | Ordinal AdaBoost2 |
| abalone | 34.59% | 26.84% | 30.83% | 39.77% | 47.34% | 39.19% | 39.89% |
| bank1 | 41.67% | 53.28% | 39.08% | 69.12% | 77.87% | 69.98% | 70.45% |
| calhousing | 36.96% | 44.66% | 37.76% | 60.07% | 63.75% | 64.13% | 58.61% |
| census1 | 41.38% | 51.22% | 37.43% | 50.76% | 56.90% | 50.70% | 53.78% |
| computer1 | 49.91% | 59.02% | 38.78% | 59.93% | 71.16% | 60.89% | 62.08% |
| computer2 | 48.04% | 57.84% | 46.62% | 64.42% | 72.91% | 63.96% | 64.36% |
| housing | 42.91% | 43.83% | 47.52% | 60.63% | 69.54% | 61.33% | 60.03% |
| machine | 40.23% | 43.05% | 40.81% | 53.54% | 59.29% | 50.27% | 50.97% |
| pyrim | 17.02% | 21.24% | 26.76% | 27.45% | 23.27% | 48.44% | 43.00% |
| stock | 62.74% | 62.85% | 58.43% | 86.81% | 90.24% | 86.27% | 85.49% |

The data sets in this paper were evaluated and compared to the results in Gutierrez et al. (2016) which evaluated 16 ordinal models based on 10-class and 5-class problems using the aforementioned 10 data sets by MAE and accuracy metrics. For comparison and simplicity, in Table 33,35,37 and 39, the best preformed model can be seen among the 16 evaluated in Gutierrez et al. (2016) for each data set, compared to the best model for 5-class problems and 10-class problems respectively. In addition, Tables 34,36,38 and 40, compared the best general model (i.e., ordinal random forest with Mode statistical function), to the best model presented in Gutierrez et al. (2016) (i.e., Gaussian Processes for Ordinal Regression) for 5-class problems and 10-class problems, respectively. For each data set the highest scores are marked with a grey background.

Table Best Accuracy results for 5 class problems

|  |  |  |
| --- | --- | --- |
| Data set | Best results Gutierrez et al. (2016) | Best results in the research |
| abalone | 49% | 50% |
| bank1 | 73% | 78% |
| calhousing | 62% | 64% |
| census1 | 68% | 57% |
| computer1 | 47% | 71% |
| computer2 | 51% | 74% |
| housing | 69% | 71% |
| machine | 61% | 61% |
| pyrim | 52% | 52% |
| stock | 90% | 90% |

Table Best models Accuracy results for 5 class problems

|  |  |  |
| --- | --- | --- |
| Data set | GPOR | Ordinal Random Forest |
| abalone | 49% | 50% |
| bank1 | 73% | 78% |
| calhousing | 62% | 64% |
| census1 | 68% | 57% |
| computer1 | 47% | 71% |
| computer2 | 50% | 74% |
| housing | 69% | 71% |
| machine | 60% | 61% |
| pyrim | 49% | 51% |
| stock | 89% | 90% |

Table Best Accuracy results for 10 class problems

|  |  |  |
| --- | --- | --- |
| Data set | Best results Gutierrez et al. (2016) | Best results in the research |
| abalone | 29% | 29% |
| bank1 | 53% | 54% |
| calhousing | 42% | 44% |
| census1 | 47% | 35% |
| computer1 | 26% | 49% |
| computer2 | 30% | 54% |
| housing | 46% | 47% |
| machine | 40% | 38% |
| pyrim | 29% | 30% |
| stock | 79% | 79% |

Table Best models Accuracy results for 10 class problems

|  |  |  |
| --- | --- | --- |
| Data set | GPOR | Ordinal Random Forest |
| abalone | 29% | 29% |
| bank1 | 52% | 54% |
| calhousing | 41% | 44% |
| census1 | 47% | 35% |
| computer1 | 26% | 49% |
| computer2 | 30% | 54% |
| housing | 46% | 47% |
| machine | 36% | 33% |
| pyrim | 22% | 26% |
| stock | 75% | 68% |

It can be seen that for the 5 class problems using an accuracy metric, the best result per data set, as well as the best model, achieved better results in 9 out of the 10 data sets. For the 10 classes’ distribution, however, the best model achieved better results in 7 out of 10 data sets.

Table Best MAE results for 5 class problems

|  |  |  |
| --- | --- | --- |
| Data set | Best results Gutierrez et al. (2016) | Best results in the research |
| abalone | 0.65 | 0.71 |
| bank1 | 0.27 | 0.22 |
| calhousing | 0.43 | 0.39 |
| census1 | 0.35 | 0.53 |
| computer1 | 0.73 | 0.31 |
| computer2 | 0.59 | 0.28 |
| housing | 0.34 | 0.33 |
| machine | 0.41 | 0.40 |
| pyrim | 0.63 | 0.62 |
| stock | 0.1 | 0.10 |

Table Best models MAE results for 5 class problems

|  |  |  |
| --- | --- | --- |
| Data set | GPOR | Ordinal Random Forest |
| abalone | 0.68 | 0.71 |
| bank1 | 0.27 | 0.22 |
| calhousing | 0.43 | 0.39 |
| census1 | 0.35 | 0.53 |
| computer1 | 0.74 | 0.31 |
| computer2 | 0.62 | 0.28 |
| housing | 0.34 | 0.34 |
| machine | 0.44 | 0.44 |
| pyrim | 0.65 | 0.65 |
| stock | 0.11 | 0.10 |

Table Best MAE results for 10 class problems

|  |  |  |
| --- | --- | --- |
| Data set | Best results Gutierrez et al. (2016) | Best results in the research |
| abalone | 1.36 | 1.54 |
| bank1 | 0.54 | 0.50 |
| calhousing | 0.87 | 0.86 |
| census1 | 0.74 | 1.17 |
| computer1 | 1.5 | 0.69 |
| computer2 | 1.23 | 0.60 |
| housing | 0.76 | 0.85 |
| machine | 0.91 | 0.90 |
| pyrim | 1.32 | 1.61 |
| stock | 0.22 | 0.19 |

Table Best models MAE results for 10 class problems

|  |  |  |
| --- | --- | --- |
| Data set | GPOR | Ordinal Random Forest |
| abalone | 1.49 | 1.55 |
| bank1 | 0.57 | 0.50 |
| calhousing | 0.92 | 0.87 |
| census1 | 0.74 | 1.17 |
| computer1 | 1.61 | 0.69 |
| computer2 | 1.3 | 0.61 |
| housing | 0.76 | 0.87 |
| machine | 0.97 | 0.93 |
| pyrim | 1.55 | 1.73 |
| stock | 0.25 | 0.20 |

It can be seen that for the 5 class problems using the MAE metric, the best result per data set outperformed in 8 out of the 10 data sets, while the best model achieved better results in 9 out of the 10 data sets. For the 10-class distribution, however, the best model achieved better results in 6 out of 10 data sets. The results from these experiments showed the superiority of the proposed ordinal models for 5 class problems for the 10 data sets compared to the 16 ordinal algorithms presented by Gutierrez et al. (2016).

## Results of ordinal decision tree-based models compared to state-of-the-art models

In this sub section, the proposed ordinal decision trees-based models were compared with the OBIG measure based on the mode statistical function with several popular non-ordinal and ordinal classifiers. Table 41 presents the AUC score for each model and data set, with the top 3 models highlighted in bold.

It can be seen that the ordinal random forest is found to be in the top 3 results for each one of the 10 data sets. The Gradient Boosting algorithm yields good results and appears to be found in the top 3 results for 7 among 10 of the data sets. In addition, the AdaBoost with both ordinal decision trees and ordinal boosting models were placed in the top 5 models for all 10 data sets. Comparing these ordinal models to the commonly used binary ordinal approach proposed by Frank and Hall (2001), shows that this methodology yielded significantly better results.

Table AUC score for top proposed ordinal models compared to state-of-the-art models.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Non-ordinal classifiers | | | | | | | | Popular ordinal | Proposed ordinal decision trees-based approaches | | | |
|  | KNN | Naive Bayes | Logistic Regression | XGB | Gradient Boosting | Decision Tree | Random Forest | AdaBoost | Ordinal Model | Ordinal Decision Tree | Ordinal Random Forest | AdaBoost1 | AdaBoost2 |
| abalone | 67.93% | 66.12% | 69.08% | 72.70% | 73.05% | 67.76% | 67.70% | 69.30% | 66.12% | 67.80% | 73.11% | 69.76% | 66.92% |
| bank1 | 91.10% | 86.03% | 81.01% | 91.37% | 91.22% | 75.67% | 83.02% | 78.54% | 87.74% | 84.73% | 91.14% | 89.01% | 87.90% |
| calhousing | 73.80% | 69.81% | 78.32% | 86.70% | 83.40% | 67.74% | 77.26% | 76.00% | 80.88% | 79.73% | 84.94% | 81.09% | 81.24% |
| census1 | 71.75% | 60.01% | 72.34% | 80.09% | 79.97% | 69.38% | 72.67% | 75.84% | 73.16% | 72.24% | 80.29% | 77.96% | 73.13% |
| computer1 | 82.69% | 79.62% | 84.41% | 87.86% | 87.13% | 75.78% | 80.52% | 81.97% | 82.26% | 78.74% | 87.01% | 85.30% | 82.91% |
| computer2 | 85.56% | 81.46% | 85.77% | 89.64% | 88.89% | 76.33% | 81.61% | 83.42% | 84.13% | 81.02% | 88.80% | 87.34% | 84.10% |
| housing | 79.73% | 77.63% | 80.72% | 85.95% | 86.36% | 80.63% | 79.57% | 80.74% | 81.73% | 80.31% | 86.03% | 79.41% | 80.76% |
| machine | 77.52% | 84.17% | 81.73% | 79.55% | 81.21% | 77.66% | 74.63% | 79.39% | 73.26% | 81.60% | 84.52% | 75.48% | 76.95% |
| pyrim | 70.22% | 62.32% | 64.86% | 66.91% | 69.20% | 66.75% | 63.41% | 64.89% | 65.75% | 65.23% | 75.27% | 67.91% | 69.60% |
| stock | 91.73% | 87.93% | 96.95% | 97.09% | 97.07% | 91.95% | 93.56% | 91.68% | 95.31% | 95.81% | 97.06% | 94.94% | 95.86% |

## Majority voting results based on non-ordinal and ordinal models

## 

This sub section applies a majority-voting ensemble approach based on ordinal and non-ordinal classifiers. The first ensemble combined only ordinal random forest with different statistical functions (Max, Min and Mode statistical functions). The second ensemble combined ordinal and non-ordinal models, 2 random forest models with Max and Min statistical functions and a XGboost model, which is shown to achieve good performance results. Table 42 presents the results of the ensemble models compared to individual classifiers which achieved the best results. It can be seen that the ensemble models based on the majority voting technique yielded better results than each individual classifier. Table 43 presents the results for each data set of the proposed ensemble models, compared to the individual classifiers of which they are composed, with the best results highlighted with a grey background. The same comparison can be seen in Table 44 for a MSE performance measure. The majority-voting models outperformed the individual classifiers in 7 out of 10 data sets. The ensemble improved the results by leveraging the strengths of each classifier individually.

Table AUC score comparison for ensemble models

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Logistic Regression | eXtreme Gradient Boost | Gradient Boosting | OrdinalModel | Ordinal Random Forest (Mode) | Ordinal Random Forest (Max) | Ordinal Random Forest (Min) |
| Ensemble1: Ordinal models | (10,8) | (4,2) | (8,6) | (10,10) | (6,5) | (7,4) | (7,5) |
| Ensemble2: Ordinal and non ordinal models | (10,8) | (7,6) | (8,7) | (7,7) | (7,4) | (7,6) | (7,6) |

Table AUC score for ensemble models and their constructors

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | eXtreme Gradient Boost | Gradient Boosting | Ordinal Random Forest (Mode) | Ordinal Random Forest( Max) | Ordinal Random Forest (Min) | Ensemble1: Ordinal models | Ensemble2: Ordinal and non ordinal models |
| abalone | 72.70% | 73.05% | 73.11% | 73.23% | 73.11% | 73.35% | 73.25% |
| bank1 | 91.37% | 91.22% | 91.14% | 91.14% | 91.24% | 91.32% | 91.38% |
| calhousing | 86.70% | 83.40% | 84.94% | 85.07% | 85.03% | 85.21% | 85.75% |
| census1 | 80.09% | 79.97% | 80.29% | 80.23% | 80.29% | 80.37% | 80.44% |
| computer1 | 87.86% | 87.13% | 87.01% | 87.19% | 87.31% | 87.48% | 87.69% |
| computer2 | 89.64% | 88.89% | 88.80% | 88.74% | 89.03% | 89.06% | 89.31% |
| housing | 85.95% | 86.36% | 86.03% | 85.98% | 86.88% | 85.84% | 86.15% |
| machine | 79.55% | 81.21% | 84.52% | 82.37% | 82.69% | 84.16% | 82.86% |
| pyrim | 66.91% | 69.20% | 75.27% | 71.88% | 71.80% | 72.91% | 71.91% |
| stock | 97.09% | 97.07% | 97.06% | 96.94% | 97.01% | 96.84% | 96.85% |

Table MSE score for ensemble models and their constructors

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | eXtreme Gradient Boost | GradientBoosting | Ordinal Random Forest (Mode) | Ordinal Random Forest (Max) | Ordinal Random Forest (Min) | Ensemble1: Ordinal models | Ensemble2: Ordinal and non-ordinal models |
| abalone | 0.412 | 0.397 | 0.400 | 0.387 | 0.397 | 0.381 | 0.391 |
| bank1 | 0.128 | 0.159 | 0.115 | 0.114 | 0.112 | 0.112 | 0.111 |
| calhousing | 0.312 | 0.362 | 0.208 | 0.205 | 0.206 | 0.203 | 0.195 |
| census1 | 0.318 | 0.343 | 0.281 | 0.283 | 0.281 | 0.280 | 0.278 |
| computer1 | 0.189 | 0.228 | 0.168 | 0.165 | 0.161 | 0.162 | 0.158 |
| computer2 | 0.161 | 0.196 | 0.143 | 0.145 | 0.141 | 0.140 | 0.137 |
| housing | 0.219 | 0.219 | 0.196 | 0.202 | 0.184 | 0.198 | 0.200 |
| machine | 0.219 | 0.258 | 0.212 | 0.222 | 0.222 | 0.205 | 0.217 |
| pyrim | 0.338 | 0.447 | 0.402 | 0.417 | 0.443 | 0.430 | 0.444 |
| stock | 0.054 | 0.071 | 0.037 | 0.039 | 0.038 | 0.040 | 0.040 |

# Conclusion & Future Work

This research proposed an extension to the objective based information-gain (WIGR) measure proposed by Singer et el., (2020), for assigning weights to different ordinal categories of the class variable, in entropy measure, with respect to a predefined objective value. In this research, an objective based information-gain (OBIG) measure was developed, with a general objective defined by a decision maker, and not necessarily restricted only to the mode category, as proposed in the previous research. The OBIG measure was used to develop ordinal decision trees based models implementing bagging, boosting and voting techniques, ordinal random forest, ordinal AdaBoost and majority voting.

The performance of the suggested ordinal models was evaluated for 10 public data sets that were used in previous research studies, and presented the superiority of the proposed models compared to other well-known ordinal and non-ordinal models based on a variety of common performance metrics. Specifically, the best individual ordinal random forest, based on a mode statistic objective, yields up to 19% improvement, compared to the conventional random forest model for all performance metrics. Similarly, the ordinal AdaBoost model yields up to 18% improvement compared to the conventional AdaBoost model. These proposed ordinal models also outperformed other ordinal models that were suggested in previews of research studies with up to 20% improvement. Finally, a majority-voting approach was investigated, that combines ordinal and non-ordinal classifiers aiming to leverage the strengths of each type of classifier. This ensemble approach achieves better performance in all indices than the best individual ordinal and non-ordinal classifiers, with improvement of 1%-3%.

Although this experiment systematically proved that the proposed ordinal models yielded better results, the experiment is performed only on 10 data sets, while some of them consisted of a small number of instances. Furthermore, it was assumed that the categories of the class variable in all data sets are equally distributed and the errors size between the values of categories are equal. Thus, future research can explore the performance of the proposed ordinal models when the categories of the classes are not equally distributed, and the errors size can be different between them. Furthermore, integrating the proposed OBIG measure into other boosting ensemble methods, such as gradient boosting or XGBoost algorithms, and development of a systematic approach for choosing the ordinal models to combine in voting techniques, are interesting issues to be explored in the future.

# References

1. Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random forests and decision trees. International Journal of Computer Science Issues (IJCSI), 9(5), 272.
2. Alpaydin, E. (2020). Introduction to machine learning. MIT press.
3. Belgiu, M., & Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. ISPRS journal of photogrammetry and remote sensing, 114, 24-31.
4. Ben-David, A. (1995). Monotonicity maintenance in information-theoretic machine learning algorithms. Machine Learning, 19(1), 29-43.
5. Ben-David, A., Sterling, L., & Tran, T. (2009). Adding monotonicity to learning algorithms may impair their accuracy. Expert Systems with Applications, 36(3), 6627-6634.
6. Ben‐David, A., Sterling, L., & Pao, Y. H. (1989). Learning and classification of monotonic ordinal concepts. Computational Intelligence, 5(1), 45-49.
7. Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford university press.
8. Cao‐Van, K., & De Baets, B. (2003). Growing decision trees in an ordinal setting. International Journal of Intelligent Systems, 18(7), 733-750.
9. Cardoso, J. S., & Sousa, R. (2011). Measuring the performance of ordinal classification. International Journal of Pattern Recognition and Artificial Intelligence, 25(08), 1173-1195.
10. Christophe, M., & Petturiti, D. (2013, August). Monotone classification with decision trees. In 8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13). Atlantis Press.
11. Costa, J., & Cardoso, J. S. An ADABOOST variant for Ordinal Classification.
12. Destercke, S., & Yang, G. (2014, September). Cautious ordinal classification by binary decomposition. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 323-337). Springer, Berlin, Heidelberg.
13. Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. The journal of machine learning research, 15(1), 3133-3181.
14. Frank, E., & Hall, M. (2001, September). A simple approach to ordinal classification. In European Conference on Machine Learning (pp. 145-156). Springer, Berlin, Heidelberg.
15. Gaudette, L., & Japkowicz, N. (2009, May). Evaluation methods for ordinal classification. In Canadian conference on artificial intelligence (pp. 207-210). Springer, Berlin, Heidelberg.
16. Gutiérrez, P. A., Perez-Ortiz, M., Sanchez-Monedero, J., Fernandez-Navarro, F., & Hervas-Martinez, C. (2015). Ordinal regression methods: survey and experimental study. IEEE Transactions on Knowledge and Data Engineering, 28(1), 127-146.
17. Hornung, R. (2019). Ordinal forests. Journal of Classification, 1-14.
18. Hu QH, Guo MZ, Yu DR, Liu JF. Information entropy for ordinal classification. Sci China, Ser F Inf Sci. 2010;53(6):1188-1200. doi:10.1007/s11432-010-3117-7
19. Hu, Q., Che, X., Zhang, L., Zhang, D., Guo, M., & Yu, D. (2011). Rank entropy-based decision trees for monotonic classification. IEEE Transactions on Knowledge and Data Engineering, 24(11), 2052-2064.
20. Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. IEEE transactions on pattern analysis and machine intelligence, 20(3), 226-239.
21. Liang, D., Tsai, C. F., Dai, A. J., & Eberle, W. (2018). A novel classifier ensemble approach for financial distress prediction. Knowledge and Information Systems, 54(2), 437-462.
22. Ling, C. X., Yang, Q., Wang, J., & Zhang, S. (2004, July). Decision trees with minimal costs. In Proceedings of the twenty-first international conference on Machine learning (p. 69).
23. Marsala, C., & Petturiti, D. (2013, July). Hierarchical model for rank discrimination measures. In European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (pp. 412-423). Springer, Berlin, Heidelberg.
24. Masetic, Z., & Subasi, A. (2016). Congestive heart failure detection using random forest classifier. Computer methods and programs in biomedicine, 130, 54-64.
25. Mourõo, R., Carvalho, R., Carvalho, R., & Ramos, G. N. (2017, December). Predicting waiting time overflow on bank teller queues. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 842-847). IEEE.
26. Potharst, R., & Bioch, J. C. (2000). Decision trees for ordinal classification. Intelligent Data Analysis, 4(2), 97-111.
27. Potharst, R., & Feelders, A. J. (2002). Classification trees for problems with monotonicity constraints. ACM SIGKDD Explorations Newsletter, 4(1), 1-10.
28. Qian, Y., Xu, H., Liang, J., Liu, B., & Wang, J. (2015). Fusing monotonic decision trees. IEEE Transactions on Knowledge and Data Engineering, 27(10), 2717-2728.
29. Sanit-in, Y., & Saikaew, K. R. (2019). Prediction of waiting time in one stop service. International Journal of Machine Learning and Computing, 9(3).
30. Sathyadevan, S., & Nair, R. R. (2015). Comparative analysis of decision tree algorithms: ID3, C4. 5 and random forest. In Computational intelligence in data mining-volume 1 (pp. 549-562). Springer, New Delhi.
31. Senderovich, A., Weidlich, M., Gal, A., & Mandelbaum, A. (2015). Queue mining for delay prediction in multi-class service processes. Information Systems, 53, 278-295.
32. Shannon, C. E. (1948). A note on the concept of entropy. Bell System Tech. J, 27(3), 379-423.
33. Singer, G., & Cohen, I. (2020). An Objective-Based Entropy Approach for Interpretable Decision Tree Models in Support of Human Resource Management: The Case of Absenteeism at Work. Entropy, 22(8), 821.
34. Singer, G., & Golan, M. (2019). Identification of subgroups of terror attacks with shared characteristics for the purpose of preventing mass-casualty attacks: A data-mining approach. Crime Science, 8(1), 1-11.
35. Singer, G., & Marudi, M. (2020). Ordinal Decision-Tree-Based Ensemble Approaches: The Case of Controlling the Daily Local Growth Rate of the COVID-19 Epidemic. Entropy, 22(8), 871.
36. Singer, G., Anuar, R., & Ben-Gal, I. (2020). A weighted information-gain measure for ordinal classification trees. Expert Systems with Applications, 152, 113375.
37. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information processing & management, 45(4), 427-437.
38. Sun, B., Chen, S., Wang, J., & Chen, H. (2016). A robust multi-class AdaBoost algorithm for mislabeled noisy data. Knowledge-Based Systems, 102, 87-102.
39. Verbeke, W., Martens, D., & Baesens, B. (2017). RULEM: A novel heuristic rule learning approach for ordinal classification with monotonicity constraints. Applied Soft Computing, 60, 858-873.
40. Vezhnevets, A., & Vezhnevets, V. (2005, September). Modest AdaBoost-teaching AdaBoost to generalize better. In Graphicon (Vol. 12, No. 5, pp. 987-997).
41. Wang, Y., Han, P., Lu, X., Wu, R., & Huang, J. (2006, October). The performance comparison of AdaBoost and SVM applied to SAR ATR. In 2006 CIE international conference on radar (pp. 1-4). IEEE.
42. Yıldırım, P., Birant, U. K., & Birant, D. (2019). EBOC: Ensemble-based ordinal classification in transportation. Journal of Advanced Transportation, 2019.
43. Zhou, Z. H. (2009). Ensemble learning. Encyclopedia of biometrics, 1, 270-273.
44. Zhu, H., Tsang, E. C., Wang, X. Z., & Ashfaq, R. A. R. (2017). Monotonic classification extreme learning machine. Neurocomputing, 225, 205-213.

**תקציר**

במחקר זה אנו מציעים גישה חדשנית לשילוב במודלים מבוססי עצי החלטות עבור בעיות סיווג אורדינליות. השיטות המוצעות לוקחות בחשבון את מאפייני הסדר של הנתונים, כמו גם את גודל טעות הסיווג הפוטנציאלית ביחס לערך מטרה מוגדר מראש טרם בניית המודלים. אנו מראים כיצד ניתן להשתמש בגישה המוצעת עבור מודלים שונים המבוססים עצי החלטות כדוגמת Random Forest ו-AdaBoost. כמו כן על מנת לשפר את ביצועי הסיווג, אנו משתמשים במודל אנסמבל כדי לשלב מסווגים אורדינליים ולא אורדינליים באופן שבו נעשה שימוש ביתרונות של כל אחד מסוגי המסווגים. כדי להעריך את הגישה המוצעת, השתמשנו במספר בסיסי נתונים ידועים שנעשה בהם שימוש בעבודות מחקר קודמות, ומצאנו כי השיטות המוצעות בעלות ביצועים טובים יותר באופן עקבי ביחס למודלים אחרים אשר מקובלים לשימוש בבעיות מסוג זה (אורדינאליים ולא אורדינאליים). לצורך הערכת הביצועים של המודלים נעשה שימוש במדדי ביצוע שונים המשמשים לבעיות סיווג למידת מכונה קלאסיות ואורדינאלית.

ֿ

**אוניברסיטת תל-אביב**

**הפקולטה להנדסה ע"ש איבי ואלדר פליישמן**

**בית הספר לתארים מתקדמים ע"ש זנדמן סליינר**

**גישות עבור מודלים מבוססי עצים לפתרון בעיות סיווג אורדינליות**

חיבור זה הוגש כעבודת גמר לקראת התואר "מוסמך אוניברסיטה" בהנדסת תעשייה וניהול

על-ידי

**מתן מרודי**

העבודה נעשתה במחלקה להנדסת תעשייה

בהנחיית פרופ׳ עירד בן גל

וד״ר גונן סינגר

שבט תשפ״א

**אוניברסיטת תל-אביב**

**הפקולטה להנדסה ע"ש איבי ואלדר פליישמן**

**בית הספר לתארים מתקדמים ע"ש זנדמן סליינר**

**גישות עבור מודלים מבוססי עצים לפתרון בעיות סיווג אורדינליות**

חיבור זה הוגש כעבודת גמר לקראת התואר "מוסמך אוניברסיטה" בהנדסת תעשייה וניהול

על-ידי

**מתן מרודי**

שבט תשפ״א