Titre: Title:	Using a Diversity Criterion to Select Training Sets for Machine Learning Models
Auteur: Author:	Kim Ton
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ton, K. (2021). Using a Diversity Criterion to Select Training Sets for Machine Learning Models [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/9902/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/9902/
Directeurs de recherche: Advisors:	Daniel Aloise, & Claudio Contardo
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Using a diversity criterion to select training sets for machine learning models

KIM TON
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Novembre 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Using a diversity criterion to select training sets for machine learning models

présenté par **Kim TON**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Michel GAGNON, président
Daniel ALOISE, membre et directeur de recherche
Claudio CONTARDO, membre et codirecteur de recherche
Quentin CAPPART, membre

ACKNOWLEDGEMENTS

I would like to start by thanking my research advisors, Daniel Aloise and Claudio Contardo for their help throughout my master's. They provided valuable advice and insights. Their support, both moral and financial, was greatly appreciated during those crazy times.

I wouldn't have been able to finish this without my colleagues and friends from the Macadamia lab and Dorsal lab. All those times spent talking, laughing, and eating either in person or online would forever be in my memories.

Finally, I want to thank my friends and family that supported me since the beginning. I wouldn't be who I am today without you guys.

RÉSUMÉ

Un modèle prédictif peut être utile seulement dans la mesure où ses prédictions reflètent la réalité. La performance d'un modèle d'apprentissage machine dépend beaucoup de l'information donnée lors de son entraînement puisqu'il se base sur cette information pour se représenter la réalité. Les méthodes de sélection de données sont des méthodes utilisées pour préparer les ensembles de données et avoir de meilleurs modèles. Elles ont comme but de réduire le nombre de données à montrer au modèle d'apprentissage machine tout en gardant la performance générale du modèle. Elles cherchent les instances aberrantes ou redondantes et les enlèvent.

Ce mémoire est un travail exploratoire sur l'utilisation de la diversité comme critère de sélection pour une méthode de sélection de données. Nous proposons une nouvelle méthode de sélection de données de type *filter* nommé MaxDivSec. Partant avec l'entièreté de l'ensemble de données, MaxDivSec résout des Maximun Diversity Problem (MDP) pour réduire d'un certain pourcentage l'ensemble d'entraînement. MaxDivSec est basée sur l'intuition que les points formant un sous-ensemble avec le plus de diversité à moins de chance d'être dans la même classe. Elle résout un Maximun Diversity Problem dans chaque classe de l'ensemble de données et enlève la solution de l'ensemble d'entraînement. La méthode cherche à réduire l'ensemble de données. Nous testons notre méthode avec deux méthodes de références : une sélection aléatoire et une sélection avec la méthode mahalanobis de détection de données aberrantes. La performance avec un classificateur de type *K-nearest neighbours* de chacune des méthodes est analysée avec un test *Wilcoxon signed-rank*. Ce test permet de déterminer si les résultats obtenus sont statistiquement significatifs et différents entre eux. Nos résultats montrent que MaxDivSec performe mieux que la sélection aléatoire et peut donc être un critère pour sélectionner des instances pour un modèle. Nous avons présenté nos résultats dans un article soumis en juillet 2021 dans le journal *SN Operations Research Forum*.

ABSTRACT

Providing the right data to a machine learning model is an important step to ensure its performance. Non-compliant training data instances may lead to wrong predictions yielding models that cannot be used in production. Instance or prototype selection methods are often used to curate training sets thus leading to more reliable and efficient models. Those methods want to reduce the training set's size while preserving the classifier's performance. They operate by removing undesirable instances raised by noise or redundancy.

In this thesis, we investigate if *diversity* is a helpful criterion for choosing which instances to remove from a given training set. We propose a new filtering method, called MaxDivSec, that solves a Maximun Diversity Problem as one of its computing steps. Our method starts with the entire training set and then reduces the training by a certain percentage.

The intuition behind the method is that the most diverse points in a class have less probability to belong together. We test our hypothesis against a random selection method and a popular outlier selection method, using benchmark datasets with different data characteristics. Our computational experiments demonstrate that selection by diversity achieves better classification performance than random selection and can hence be considered as an alternative data selection criterion for effective model training. Those results are shown in an article submitted for the *SN Operations Research Forum* journal in July 2021.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

ML	Machine Learning
KNN	K-nearest neighbours
MDP	Maximun Diversity Problem
SVD	Singular Value Decomposition
RMSE	Root Mean Squared Error

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Classification problems are a core subject in artificial intelligence and Machine Learning (ML). ML is the science of finding patterns within information to be able to make predictions. The information that we want to analyze is called the *raw data*. It is collected prior to the ML exercise and is preserved in a matrix of size $m \times n$. In classification, a label must be decided for a new data instance based on the information gained from previously seen data instances. The label represents the class to which the instances belong. A simple classification problem would be to separate balls into two classes: small ones (5cm and less) and big ones (10cm and more). A data instance in this case is a ball among the ones available. The features of an instance are all the characteristics we have about it. In this case, it could the ball's color, the ball's weight, the ball's material, and the ball's diameter. The raw data would then be all those characteristics for 1000 balls forming a matrix of size 1000 x 4. A ML model could then generalize that any diameter smaller than a certain value is considered as part of the small class. Thus. it finds a separation limit, commonly denoted threshold, between the two classes. If that assumption is correct, it will then be able to classify new balls in the correct class. Other examples of classification problems can be something as simple as separating various iris species or as complex as determining if a patient has a cancer tumor.

Before an ML model can make predictions, it has to be trained. From the raw data, a training set is formed and then given to the ML model. The simplest training set encompasses all the available data. Indeed, the quality of an ML model depends on the quality of its training set. If it is given erroneous information, the model will be susceptible to making wrong predictions because of prior wrong assumptions. Directly giving the raw data as a training set is not optimal. That is why preprocessing methods are crucial for an ML model. Those methods transform the raw data before the model makes use of it. There exist various transformations that can be done on the data to make sure of its quality. A common one is to find outliers and remove them from the training set. Outliers, also called noisy instances, are erroneously labeled instances that are too different from the rest of the data. An example of an outlier would be an orange among apples. Another transformation is to reduce the number of features. The idea is that not all the characteristic matters for the problem and by removing some, we simplify the problem for our model making predictions easier. For the example above about balls of different sizes, we know that only the diameter is important, and all the other features can be ignored. Real ML problems are more complex and as such seeing which features should be removed is not trivial. Feature selection methods propose ways to determine which features are important. Another preprocessing method consists in trying to

reduce the number of instances provided to the model. These methods are called *instance selection methods*. They use a selection criterion to either select which instances from the training set to keep or to remove. Their goal is to reduce the training set without affecting the model's capacity to make correct predictions. The idea behind them is that noisy and redundant instances can be removed from the training set without prejudice, or maybe even improve a model's classification performance. Those instances are undesirable because they may confuse the model, then leading to the generalization of a reality that is different from the truth. As such, they don't help the model gain additional information about the problem. For example, if a big ball is labeled as small and provided to the model, the obtained threshold for the balls will tend to be higher than the actual necessary value. A by-product of reducing the size of a training set is the reduction of the required time for training ML models, which is a bottleneck for attaining good classification rates in some real-life applications [1].

We want to explore the possibility for instance selection methods by using diversity as a criterion. We define diversity as the observed variation among data instances in a set. That variety is quantified by their dissimilarities [2]. Since diversity is linked to the dissimilarity between instances, it can be used to find the instances that fit less together in the current class. By removing the most diverse instances from the class, we hope to make the class more homogeneous and easier to generalize for our model. A classical problem that tackles diversity is the Maximun Diversity Problem (MDP).

The MDP was first formulated by Kuby in 1987 and is an NP-hard problem [3]. The problem has attracted attention because of its various applications for real-life problems. It was first used to help place competing stores in a city, placing trash/pollutant storage as to not concentrate exposure in an area of the town [3, 4]. It is now applied in a broader range of domains like in diversity decision for ecosystems re-population or genetic research on virus and cures [5–9].

In the MDP, one chooses a subset S of m points from a set U of n points that maximises the sum of pairwise distances between all the points in S. Using a distance matrix $D = \{D(i,j) : 1 \leq i,j \leq n\}$ defined such that $D(i,j) \geq 0$ for every $1 \leq i,j \leq n$ and $D(i,i) = 0$, the problem can be mathematically described as:

$$\begin{aligned} &\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} D(i,j) x_i x_j \\ &\text{subject } \sum_{i=1}^{n} x_i = m \\ &\qquad x_i \in \{0,1\} \qquad\qquad \forall i = 1, \ldots, n. \end{aligned} \tag{1.1}$$

1.1 Elements of the problem

With the advancement of data science and ML, data is being generated at a frightening speed. The use of all that available data is generally inefficient and costs lots of resources and time. Usually in ML, preprocessing methods are used to clean and prepare the data before the training phase. Giving the raw data directly is often detrimental to the model‘s performance because of missing or invalid values. Normalizing the data is very common to reduce the range of features and make it easier for the model to learn. Some methods can also reduce the data‘s size by removing useless or redundant information. There are two ways to reduce the data's size: by cutting it in the number of features or by cutting the number of instances. Feature selection methods fall under the first category while instance selection methods and outlier detection methods fall under the latter. While both categories remove information deemed useless or redundant, the first one cuts in the number of columns and the second cuts in the number of rows given to the model. It means that feature selection methods focus on features while instance selection methods focus on instances. This master's thesis explores the idea of using maximum diversity as a criterion for data instance selection methods.

1.2 Research objectives

The main objective of this master's thesis is to explore the feasibility of using diversity as a criterion for an instance selection method. To do so, we propose a new instance selection algorithm. Our hypothesis is that diversity is a good criterion for instance selection because we analyze selected instances as a subset. For each class, our proposal removes the subset of instances with maximum diversity. Thus, our criterion focuses on the diversity of the chosen subset to be removed. It allows us to make decisions at a subset level instead of using an individual selection criterion (e.g. via outlier scores).

The contributions from this thesis are:

1. We propose a new instance selection method using maximum diversity as an instance selection criterion. (`MaxDivSec`)

2. We explore the feasibility of using diversity as a criterion and prove our research hypotheses described below:

    - R1: Is diversity a valid criterion for instance selection?
    - R2: Is diversity better than other instance selection criteria used in the literature?

The contributions from this thesis were presented in an article submitted to the *Operations Research Forum* journal in July 2021. The result of the submission is expected to be known after the thesis's submission.

1.3 Thesis outline

This master's thesis is followed by a literature review on instance selection methods in Chapter 2 and by an explanation of the approach done for this research in Chapter 3. More details are given on the direction of the research and its objectives. After that, the article written on our new method for the *Operations Research Forum* journal is given in Chapter 4. The article contains the results obtained from this research. After that, a short general discussion in Chapter 5 on the relation between the article and the thesis. The paper concludes in Chapter 6 with a summary of the results and limitations of the research and propositions for future works.

CHAPTER 2 LITERATURE REVIEW

This chapter introduces the state of the art on instance selection methods in predictive analysis. The section 2.1 explains basic concepts about Machine Learning. The section 2.2 focus on existing instance selection methods. This allows us to propose a solution that is new and potentially better. Section 2.3 gives a more in-depth explanation about classification, and especially about the K-nearest neighbor model, which is used in our experiments. Section 2.4 describes the distance metric used for our experimentation. Finally, the chapter ends with a discussion about our findings from the literature review 2.5.

2.1 Machine Learning pipeline

Machine Learning (ML) is a subsection of the artificial intelligence field. It comprises models able to extract patterns from raw data [10]. ML models use a training set to learn and to then be able to either predict an outcome or make a decision. In our case, we want models to label various data instances with the proper classification.

An ML model needs data to learn. Each element of the data is called an instance. An instance has attributes or features with which the model learns information about the data. While ML pipelines have become more elaborate nowadays due to more complex models and data, it can be resumed in 3 principal stages: preprocessing, training, and prediction/decision stages [11], as shown by Figure 2.1. The preprocessing stage is where the data is prepared for the model. The data is transformed and cleaned using various methods. The training phase is when the ML model uses the given data to generalize rules to make predictions. The prediction/decision stage is when the ML model's ability to generalize is tested with instances that the ML model never saw before.

While training a model, the data is often split into two sets: the training set and the test set. The train set is used to prepare the model while the test set is used to simulate how the model will perform once deployed with unseen instances. Both sets go through the preprocessing stage before being presented to the model. The idea is to use those sets to

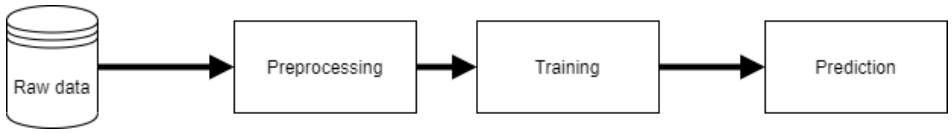


Figure 2.1 ML basic pipeline

choose the correct parameters for the model. We give the training set to the model and then check the performance with the test set. The ML model is trained multiple times with new parameters. By comparing the performance results, the best parameters can be chosen and kept for the model to be used in production. That procedure is called cross-validation [12]. Typically, the training set is further split to create a validation or pruning set [13]. That third set is used instead of the test set to measure the model's performance. By choosing the parameters with the performance obtained with the validation set, we keep some data unknown for a fair last test for the trained model. That allows us to avoid overfitting the model on the raw data that we have and to have a better idea of the model's abilities to generalize.

2.2 Instance selection methods

Instance selection methods are methods applied to data during the preprocessing phase of an ML pipeline. Their goal is to reduce the amount of data provided to an ML model without compromising the quality of the dataset. As such, the method targets redundant or noisy instances to be removed [14, 15]. For a training set T, the method wants to produce a new reduced set U such as $U \subset T$.

2.2.1 Characteristics

Instance selection methods can be classed using three main characteristics: the direction of search, the type of selection, and the evaluation of search [15].

An instance selection method's direction of search is how the method builds the selected subset, which can be either incremental or decremental. An incremental search starts with an empty subset and then adds the instances that fit the criterion one by one. A decremental search starts with the entire set and removes the instances that fit the criterion to form a subset. Finally, a mixed search starts with a valid subset and then adds or removes instances depending on the used criterion to improve the subset.

The type of points the method wants to keep is a characteristic called the type of selection. A condensation method will focus on keeping the points that are close to the decision boundaries. Decision boundaries are lines made by classifiers in the dataset to decide for a classification. By only keeping the decision points, the decision boundaries are not affected and the inner points can be discarded without much impact. An edition method focuses on removing decision points and keeping the inner points. That makes the border smoother for the classifier and makes the separation clearer between the classes. Hybrid methods allow

the removal of both decision points and inner points depending on the criterion used.

The last characteristic, called evaluation of search, refers to the method's search framework. Those frameworks can be classified into two categories: Wrapper methods and filter methods. A simple decremental algorithm for a wrapper method and a filter method are presented in Algorithm 1 and 2, respectively. We can see that their difference lies in what is used as a condition to remove an instance. Wrapper methods use a criterion based on accuracy. They proceed by optimizing a hidden classifier, iteratively removing instances that do not impact the classifier's accuracy. Conversely, filter methods do not optimize a hidden classifier. Data instances are filtered if they met a certain criterion established as a function of the instance's properties.

Algorithm 1: `Wrapper method`

Input: T: labelled dataset of dimension $m \times n$

$U \leftarrow T$
Let *accuracy* be the accuracy with T
for $c = 1, \ldots, m$ **do**
  if *The accuracy for* $U \setminus \{c\} \geq$ *accuracy* **then**
    $U \leftarrow U \setminus \{c\}$
  end
end
return U

Algorithm 2: `Filter method`

Input: T: labelled dataset of dimension $m \times n$

$U \leftarrow T$
for $c = 1, \ldots, m$ **do**
  if *The criterion* X *is met with data instance* c **then**
    $U \leftarrow U \setminus \{c\}$
  end
end
return U

2.2.2 Related works

Wrapper methods were the first type of methods developed for instance selection. Most of the existing methods are based on the K-nearest neighbours classifier to evaluate the accuracy obtained with the training set. As such they have better results with a ML model that also uses the K-nearest neighbours rule. The simplest one was proposed by Hart in 1968 [16].

It is an incremental condensation method that takes one instance in each class randomly to create the initial training set U and then tries to classify T using U. If an instance is misclassified, it is added to U. That method is called the Condensed Nearest Neighbour (CNN) [17]. The Selective Nearest Neighbour (SNN) method was proposed by Ritter et al. in 1975 as a variation on CNN [18]. The SNN has an extra condition where all instances in T have their closest neighbor of the same class in U. It means T can be correctly classified with 1NN using U. Those methods are weak to noisy instances since they would be misclassified and then kept in U. The method can create multiple solutions depending on the order in which the instances are presented.

Another known method was proposed by Wilson in 1972 called Edited Nearest Neighbour (ENN) method [19]. It is a decremental edition method that removes instances that are different from the majority of their closest neighbours. The method usually checks the 3 nearest neighbours to make the decision. This method starts with $S = T$ and then focuses on removing noisy instances. Another version of that algorithm is the Repeated Edited Nearest Neighbour which applies ENN until all instances in U have the same class as the majority of their neighbours.

Ahal et al. proposed instance-based learning algorithms (IB) in 1991 [20]. The first algorithm is called the IB1 algorithm and isn't considered as an instance selection method [20]. It is the base rule for the criterion of the IB2 and IB3 instance selection methods. They use the concept of similarity to classify instances without generalizing them. They go through each instance one by one and make a decision based on the previously seen instances. IB2 and IB3 are considered incremental hybrid methods since they don't focus on a particular type of instance to keep but just on the classifier‘s accuracy. IB2 selects all the instances misclassified by the 1NN method with the current U. IB2 is weak towards noisy instances and to fix that IB3 was proposed. This method adds an extra element to the decision phase. It tracks the number of good and bad classifications for each instance seen after it was added to the training set. That gives an idea of their performance to classify, and the more performing ones are kept. It prevents noisy instances from being added to U. These methods are also dependant on the order in which we present the instances and can create multiple solutions to the problem.

Decremental Reduction Optimization Procedure methods (DROP) are instances selection methods based on the concept of *asociates* proposed by Wilson and Martínez in 2000. All instances that have i among their K nearest neighbour are considered *associates* of i. They are decremental hybrid methods and use the associates to decide which instances to remove from T to create U [21]. The simplest is DROP1 which removes all i for which all its

associates in U can be correctly classified without it. This method drops noisy instances but can also keep them if their neighbours were dropped first. To prevent that, DROP2 check the associates in T instead of U. DROP3, DROP4 and DROP5 apply a filter to discard noisy instance or nearest enemies to smooth the decision boundaries before applying DROP2.

Filter methods are instance selection methods that don't use a performance score as the criterion. Filtering methods are more expensive in terms of computation power than wrapper methods but they are more flexible since they do not limit themselves with the K-nearest neighbours rule. Newer propositions are often filter methods since multiple wrapper methods were proposed already. Those methods use the concept of border and interior instances in their criterion. An instance i is considered a border instance when it is the nearest neighbour of an instance from another class. All other instances are considered interior instances.

The Mahalanobis outlier detection method can be considered a filter instance selection method since it doesn't use a classifier performance to decide which instances to be removed. It uses the Mahalanobis distance as weight and removes the highest using a threshold. This approach was explored by many methods with different weight formulas like the Weighting Prototypes (WP) by Parades and Vidal (2000) [22] and the Prototype Selection by Relevance (PSR) by Olvera-López et al. (2008) [23]. They are considered decremental edition methods. WP used a gradient descent approach to calculate a dissimilarity score for each instance using both nearest neighbours and nearest enemies. It removes the instances that have scores above the threshold. PSR is a method that is a little more complex. It calculates a similarity score using the Heterogeneous Value Difference Metric (HVDM) [24] for each instance. This distance function can be used with numeric, non-numeric, and missing features. PSR starts by selecting the p most relevant instances per class by using the highest similarity scores. For each relevant instance, it finds the nearest neighbour in all the other classes and adds it as a border instance. Those two types of instances form the final training set.

Another explored direction with filter methods was to use clustering to select the instances. The intuition was to form p clusters with the training set and then to choose only the center of each cluster [25]. The Generalized-Modified Chang Algorithm (GMCA) proposed by Mollineda et al. (2002) [26] adds another step and merges each pair of nearest clusters with the same class. It then selects the center of those newly merged clusters.

Methods can also invent their own criterion like a method called Pattern by Ordered Projection (POP) [27] proposed by Riquelme et al. (2003). It is a decremental condensation method based on the concept of *weakness*. The *weakness*(i) of an instance i is defined as the number of times where i isn't a border instance in a class across its attributes. The maximal weakness is equal to the number of attributes and represents an interior instance

to be removed.

2.3 Classification problems

Classification problems are ML problems where the model has to decide in which class each instance in the data belongs. The model can know how many classes there are or deduce those classes by itself. A classification problem with 2 classes is a special case called a binary problem. One class is called the positive class and the other is the negative class. Another particular case called imbalanced problem is when the majority of the instances are concentrated in a few classes. An unsupervised class model does not have access to the classes of the training set. It must deduce the classes based on the similarities between the instances. To help it, the number of classes expected is often given. A supervised classification model has access to the classes of the instances in the training set and it does not need to deduce the classes.

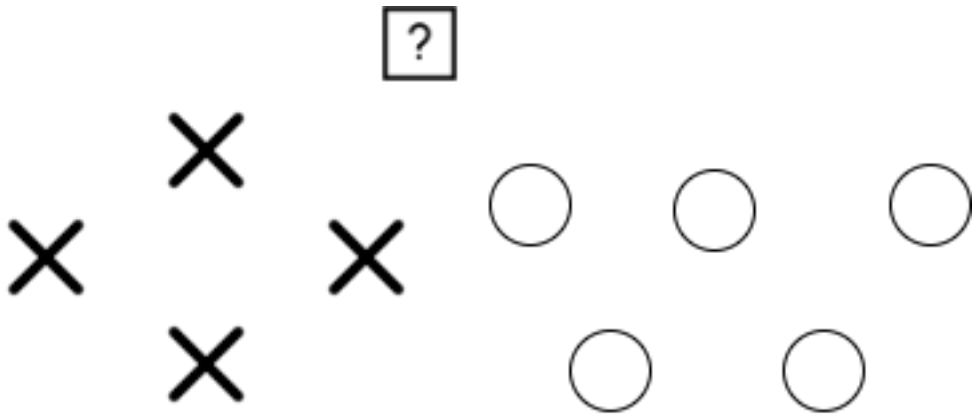
2.3.1 K-nearest neighbours model

K-nearest neighbours (KNN) is one of the simplest ML models used for supervised classification. It is used in multiple domains like computer vision problems [28], medical diagnostic [29] and recommendation systems [30]. KNN is a classifier that proved its usefulness and is common to the ML world.

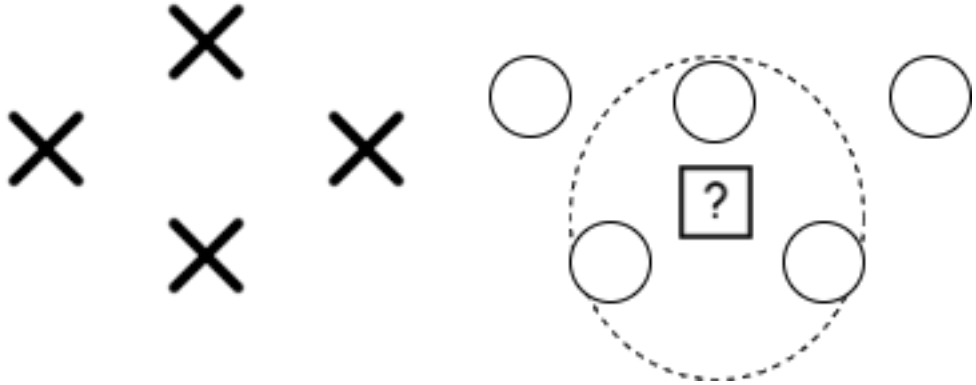
To train an KNN classifier, we have to provide a labelled training set. KNN keeps then all those instances in memory. During the prediction phase, it finds the K nearest instances to the new unlabelled instance among the labels of the instances of the training set. The evaluated data instance is then given the majority's class among its closest neighbours [16]. This is called a voting phase where each neighbour votes for their own class. KNN is a simple but effective non-parametric classifier. Depending on the distance metric used, the classifier may find different neighbours. Figure 2.2 shows simple examples of how a KNN model predicts a class for which a label is not known.

The most common distance metric for KNN is the Euclidean distance. Depending on the data, other metrics could be more adapted to calculate the distance between the instances. An example is the Euclidean embedding that transpose instances in an Euclidean space to facilitate their comparison [31–33].

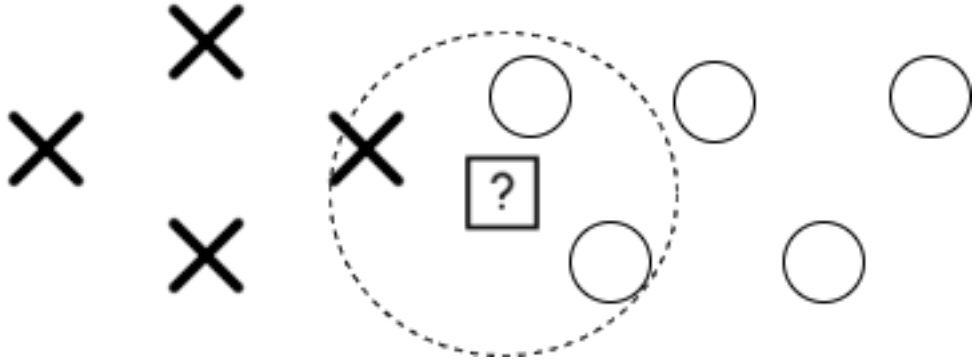
The KNN model is known as a lazy-training model. It only keeps in memory the previously seen instances and doesn't create a generalization of the reality as explained previously [16]. That is why it is really important to give good instances to this type of model. The quality of



(a) This is a fictional dataset with two classes: the cross and round classes. We want to predict the class of the instances marked with a ?.



(b) With $K = 3$, we can see that the instances inside the dotted circle are the closest neighbours. The new instance would be predicted as a round class since all the neighbours are from that class.



(c) With $K = 3$, we can see that the instances inside the dotted circle are the closest neighbours. The new instance would be predicted as a round class since the majority of its neighbours are from that class.

Figure 2.2 Examples for the KNN model

the prediction depends on the quality of the instances given to the model. The model does not use any measure to temper the influence of outliers or noisy instances. The less neighbours the classifier checks, the more impact the instances have on the prediction. However, a higher K could consider instances that are too far to be pertinent for the decision. That is why choosing the K used for a model is important and is usually done by cross-validation. Another thing to consider is the possibility of having a tie during the voting phase. Different strategies for breaking ties exist. The *scipy* implementation takes the first class seen among the tied class. Another way to break the tie is to take the class of the nearest neighbour. To avoid ties, it is suggested to use a K that is a multiple of $\sqrt{n}$ where n is the number of instances in the training set [34].

2.3.2 Models' performance

An ML model's performance reflects the model's ability to find patterns in the data. Several metrics are used to evaluate a model's performance depending on the type of problem. The performance is also measured to help decide which parameters are better for the model. To calculate the performance, we need the ground-truth classes for all the instances given to the model to compare them with the model's prediction. A *true* case happens when the model predicts the correct class while the *false* case happens when the model predicts another class than the expected one. In a binary classification problem, there exist four possible cases for a given prediction: a True Positive (TP), a False Positive (FP), a True Negative (TN) and a False Negative (FN) [35]. We will use those concepts to explain the various performance metric used. A simple and intuitive one is the accuracy metric. It is simply the number of correct predictions on the total number of predictions made [36]. It can be mathematically described this way:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

The accuracy metric can also be generalized for k classes as

$$Accuracy = \frac{\sum_{c=1}^{k} T_c}{\sum_{c=1}^{k} T_c + \sum_{c=1}^{k} F_c} \tag{2.2}$$

where T_c is the number of TP for class c, and F_i the number of FP for that same class.

The accuracy metric is not recommended for imbalanced binary problems because the model is not penalized for only predicting the majority class. The F1-score is better in those situations because it uses the harmonic mean between the precision and the recall to evaluate

the performance [35]. The precision is the ratio of correctly labeled instances from the minority class to the total number of instances labeled by the model with that class. The recall is the ratio of correctly labeled instances with the minority class on the total number of instances belonging to that class. The recall and the precision of a model are computed as:

$$recall = \frac{TP}{TP + FN} \qquad precision = \frac{TP}{TP + FP}. \tag{2.3}$$

The F1-score is finally calculated as the geometric mean of both measures:

$$\text{F1-score} = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{2.4}$$

This score penalises both only predicting the majority class and never predicting the majority class.

For ordered classes, the Root Mean Squared Error (RMSE) score is used [37]. Like the name says, this metric calculates the root of the mean of differences between the expected value (y) and the predicted one (y'). The formula with n as the number of predictions is given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2} \tag{2.5}$$

It gives an idea of by how much the model is off compared to the expected values.

2.4 Distance metrics

Distance metrics are formulas to calculate dissimilarity between data instances. The idea is that distance is a good measure for the difference. The farther instances are from each other, the more different they are. T Different distance measures may lead to different MDP solutions as different instances might be selected to the composed subset. Distance measures are equally important for an KNN classifier which uses distances to identify the K nearest neighbourhood of a data instance. Multiple metrics exist but the most popular for numerical data are the Euclidean distance and the Mahalanobis distance.

Euclidean distance

The Euclidean distance is the default way of calculating a distance. It is the shortest straight line between two objects. For two vectors with n attributes, the Euclidean distance is calcu-

lated as:

$$d(\vec{q}, \vec{j}) = \sqrt{\sum_{i=1}^{n}(q_i - j_i)^2}. \tag{2.6}$$

Mahalanobis distance

The Mahalanobis distance is another well-known distance metric. It is the distance between a point and a distribution. That is why this metric is often used to see if a point fits among a set. For two vectors and Σ the covariance matrix computed from the data distribution, the Mahalanobis distance is calculated as:

$$d(\vec{q}, \vec{j}) = \sqrt{(\vec{q} - \vec{j})^T \Sigma^{-1} (\vec{q} - \vec{j})}. \tag{2.7}$$

2.5 Critical discussion

After the literature review, we decided to do an exploratory work and proposed a new method based on a new instance selection criterion. Wrapper methods are often weak against noisy instances. Also, they often depend on the order in which the instances are presented. To avoid those problems, we chose a filter method with diversity as a criterion.

In our evaluation, using diversity as a criterion for an instance selection method deserved investigation. Diversity would allow our instance selection method to not be impacted by the order of instances since it is a diversity that quantifies the instance from its selected subset. The diversity criterion is also global in the sense that the most diverse subset is evaluated in conjunction with the whole dataset.

Our proposed method can be categorized as a new decremental hybrid filter method. It is hybrid because the most diverse subset can be composed of both border and interior data instances.

Finally, We have also decided to use KNN as the underlying classifier for our tests since it is simple and relies on dissimilarities only.

CHAPTER 3 RESEARCH APPROACH AND OVERALL ORGANIZATION

In this chapter, we will present how we conducted this research. The first section talks about the methodology and coding environment. The next section talks about the code used for this research.

3.1 Methodology

The goal of this research is to test a new instance selection method. To do so we followed this methodology:

1. Find datasets with different characteristics
2. Prepare the data from each dataset
3. Run our method and the baseline methods on each dataset
4. Compare and analyse the results

3.1.1 Datasets

To simplify the preprocessing needed, we decided to use classification problems with numerical attributes only. We chose datasets of various sizes and dimensions to see how our method would react to those conditions. We decided on 8 datasets from the UCI Machine Learning Repository[1]: Iris [38], Seeds [39], Dermatology [40], Ionosphere [41], Breast Cancer Wisconsin (original) [42], Mammographic masses [43], Contraceptive methods [44] and Abalone [45].

3.1.2 Preprocessing

We kept this step the simplest possible. We remove features that are categorical and then normalised the data on each feature. We then proceed to do 10 5-fold cross-validation to create 50 different pairs of training/test sets. Those pairs are used to test the methods in the next step.

[1] https://archive.ics.uci.edu/ml/datasets.php

3.1.3 Running instance selection methods

In this step, we run our proposed method (`MaxDivSec`) and two baseline methods on each pair of training/test sets. `MaxDivSec` chooses the most diverse subset to be removed from each class. We also tested keeping the most diverse subset instead, but it was not as performing.

The baseline methods are run to help us analyse the results obtained by our method. We chose to use the `Random` method and the `Mahalanobis` method. The `Random` method consists of randomly choosing instances to be removed from the training set. The `Mahalanobis` method is based on the Mahalanobis outlier detection method. It is a well-known method to remove outlier instances using the Mahalanobis distance as a criterion to discard overly different instances.

During the tests, instance selection methods reduced the training set by 12.5%, 25%, and 50%. We recorded the performance with the KNN classifier to test the selection using the 3, 5, and 10 nearest neighbours. We also recorded the performance of the classifier with the full training set.

3.1.4 Analysis

The analysis is done first by a visualization of the results followed by a statistical test to prove that our results are statistically relevant. We chose box-plots and line plots to showcase our results and decided on the Wilcoxon signed-rank test [46] to compare the different results.

3.2 Code used

During the research both existing code and new code were used. We didn't want to code everything from scratch. All the code created by us can be find in this Git repo: `https://github.com/ktton/MaxDivSec`.

3.2.1 Coding environment

We coded this experiment in Python. The tests were run on a personal laptop dual booting Windows and Linux with the following specs:

- Microsoft Windows 10 Home or Ubuntu 20.04.2 LTS
- 8GB of RAM
- processor Intel Core i5-6200U CPU @2.30GHz (2 physical cores, 4 logical cores)

3.2.2 Principal python libraries

Python has multiple libraries specialized for machine learning projects. We used a few of them for our experiment. The pandas[2] and numpy[3] libraries were used to manipulate the data. For the various performance metrics needed (Accuracy, RMSE and F1-score) for our tests, we used sklearn[4] implementations. The Wilcoxon statistical test comes from the scipy library[5].

3.2.3 MaxDivSec

Our method needs to solve a MDP to choose which points to remove from the training set. We decided to use an already implemented heuristic solver called OBMA [47], implemented in C++. This heuristic needs a distance matrix, the number of elements to remove, and the total number of elements as inputs. The various distance matrix was constructed in Python using sklearn's Mahalanobis distance implementation. OBMA is an algorithm that combines the concept of Opposition-Based Learning (OBL) with the memetic search framework. A memetic algorithm starts with a set of solutions called the initial population. From those solutions, new solutions called offspring are created via crossover or recombination. Those offspring are then optimized. They join the population depending on the population management strategy. Opposition-based learning takes a candidate solution but also considers the opposite solution at the same time. OBMA uses an OBL to initialize the population with quality solutions. The offspring are then created using two random parents and a backbone-based crossover operator. OBMA then uses the Opposition-Based Double Trajectory Search Procedure (ODTS) to look around the offspring and its opposite solution to improve them. The ODTS is an improved constrained neighborhood tabu search. Finally, a rank-based pool updating strategy is used to decide if the offspring should be added to the solution. The algorithm repeats until the time limit is reached.

3.2.4 Test pipeline

All the code to run the various methods on the data and collect the performance scores was written by us. We used pandas, numpy and sklearn to facilitate the various ML phases. Those phases are:

1. read the preprocessed data.

[2]https://pandas.pydata.org/
[3]https://numpy.org/
[4]https://scikit-learn.org/stable/
[5]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html

2. run the three instances selection methods to create training sets.

3. record the performance with each reduced training set.

To analyze the performance scores obtained, we used scipy but also plotly[6] to create figures to visualize the results.

[6]https://plotly.com/python/

CHAPTER 4 ARTICLE 1: ON REMOVING DIVERSE DATA INSTANCES FOR TRAINING MACHINE LEARNING MODELS

Authors: Kim Thuyen Ton, Daniel Aloise, and Claudio Contardo

Submitted to *Operations Research Forum* journal in July 2021.

4.1 Introduction

Machine learning (ML) has often been perceived as requiring the largest possible amount of data to gain accuracy in predicting a behavior. Typically, there are three stages for a ML model: preprocessing, training and decision/prediction [11]. During preprocessing, the provided training set might be transformed before being fed to the ML model. In the sequel, during the training stage, the model processes the training set to generalize rules and formulas for prediction with a minimum amount of classification errors. Finally, at the prediction stage, new unlabelled data instances are given to the ML model which must predict a class or a value for them.

Nowadays, several preprocessing methods exist to ensure that only the right data is given to an ML model – a concern that accompanies the ML field since its origin [48, 49]. For example, a model that overfits or underfits the training data will result in poor predicting capabilities as they lose their ability to generalize over unseen data [50]. In addition, being able to reduce the amount of data needed to correctly train a ML model is crucial to speed up its training process and save memory resources.

Preprocessing techniques can be mainly categorized into feature selection, instance selection, and outlier detection methods [48, 49]. All of them seek to decrease the amount of data fed to a classification model. Feature selection methods reduce the training dataset by decreasing the number of used features, therefore the dimension of the data. Those methods weight the features in order of relevance and remove the least important ones [51]. Both outlier detection methods and instance selection methods work by reducing the amount of data instances. A method can either focus on removing noisy instances, superfluous data instances or both. Noisy or outlier data instances deteriorate the performance of classifiers when added to the training set while superfluous instances do not impact the performance when removed [15]. Outlier detection methods, as the name indicates, focus on removing outliers from the dataset [52].

The goal of an instance selection method is to speedup the model training by reducing

the size of the training set without impacting the model's performance [14, 15, 53]. An instance selection method can either start with an empty training set and add data instances, or start with all the data then remove instances. The selection criterion is usually based on a performance metric or a selection formula. With a metric performance, the methods reduce the training set as long as the classification performance stays above a predefined threshold [14]. With a selection formula, the stopping condition is typically defined by the user, e.g. number of needed instances, logical tests, etc. Multiple criteria can be combined together to achieve a more complex method [54].

In this paper, we investigate if *diversity* can be used as an effective criterion for removing instances within selection methods. Our concept of diversity is related to variety among the data instances, which is quantified by the observed dissimilarities among them [2]. Our research hypothesis is that the removed data instances are diverse, representing instances less likely to belong together to the same class. Thus, given that one decides to reduce the training size of a ML model, these data instances are rather selected to be suppressed.

We test our approach on classifying eight different benchmark datasets, comparing it with two baseline methods. The first one selects data instances for suppression completely at random whereas the second consists of the classical Mahalanobis outlier detection method [15, 55].

The paper is organized as follows. In Section 2, we present the *maximum diversity problem* which is optimized to decide the data instances to be removed from the available training set. In section 3, we explain our instance selection method based on maximum diversity. Section 4 describes the experimental methodology used to test our research hypothesis. In Section 5, we present and discuss the performed computational results. Finally, in Section 6, we present our concluding remarks.

4.2 The maximum diversity problem

Given a set of n data instances $U = \{u_1, \ldots, u_n\}$ for which a symmetric dissimilarity matrix $D = \{d_{ij} : 1 \leq i, j \leq n\}$ is defined such that $d_{ii} = 0$ and $d_{ij} \geq 0$ for every $1 \leq i < j \leq n$, the *maximum diversity problem* (MDP) consists of selecting a subset $P \subset U$ of size $p < n$ such as the sum of dissimilarities between the elements of P is maximum. The problem is formalized as:

$$\begin{aligned} &\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \\ &\text{subject } \sum_{i=1}^{n} x_i = p \\ &\qquad x_i \in \{0,1\} \qquad \forall i = 1, \ldots, n. \end{aligned} \tag{4.1}$$

The MDP arises in many real-life applications. For example, in facility location, one may be interested in locating competing stores in a city as far as possible, or to place trash/pollutant storage as to not concentrate exposure in one area of the town [3,4]. The MDP is also applied in biology for deciding about ecosystems re-population or for genetic engineering to produce more resilient plants [5–9], or for product design where companies want to have products that are different from their competitor [56]. The problem was shown to be NP-hard by Kuo et al [2].

Several exact and heuristic methods have been proposed in the literature to solve the MDP [4,57,58, e.g.]. The state-of-the-art exact method for the MDP is due to Marti et al. [59] who proposed a branch-and-bound able to optimally solve medium-size instances with $n = 100$ in 1 hour of CPU time. Regarding heuristics, Marti et al. (2021) [60] have very recently performed an exhaustive comparison of state-of-the-art heuristics on the MDPLIB 2.0 - Maximum Diversity Problem Library available at https://www.uv.es/rmarti/paper/mdp.html. Among the compared methods, the OBMA method of [47] emerges as the best heuristic.

4.3 Instance selection by maximum diversity

Using the MDP as underlying optimization model, we propose a new instance selection method which removes p data instances from the training set. The so-called *Max Diversity Instance Selection Method* (`MaxDivSelec`) is described in Algorithm 3. `MaxDivSelec` proceeds by removing a total of p data instances from the classes of the training dataset. For that, it solves a MDP in each class. The algorithm starts by initializing the training set with all labelled data instances (line 1). After that, the algorithm iterates (lines 2-8) over each class label $c = 1, \ldots, k$ of the training dataset. In line 3, the data instances of class c are isolated in $X_c \subseteq X$, and then the covariance matrix X_c is computed in line 4. Then, in line 5, a matrix D_c of distances is computed for each pair of instances in class c. In our case, $D = (d_{ij})$ are computed as Mahalanobis distances, i.e.,

$$d_{ij} = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}. \tag{4.2}$$

We note that Σ is approximated by singular value decomposition (SVD) factorization if it is singular [61]. In the sequel, a MDP solver – in our case OBMA [47] – is called to solve an MDP problem for D_c, selecting the $\overline{p}_c$ points of maximum diversity in class c. More details on how $\overline{p}_c$ is computed are given in section 4.4.3. The algorithm then removes the selected data instances from the training set in line 7. Finally, the reduced training set T is returned in line 9.

Algorithm 3: `MaxDivSelec`

Input: X: labelled dataset of dimension $n \times s$,
$\overline{p}$: array of dimension $1 \times k$ with the number of instances to be removed per class

1 $T \leftarrow X$
2 **for** $c = 1, \ldots, k$ **do**
3 Let $X_c \subseteq X$ be the matrix of dimension $n_c \times s$ composed by the data instances of class c in X
4 Compute the covariance matrix Σ_c of X_c
5 Compute a distance matrix D_c of dimension $n_c \times n_c$
6 $R \leftarrow$ `SolveMDP`$(D_c, \overline{p}_c)$
7 $T \leftarrow T \setminus R$
8 **end**
9 **return** T

Figure 4.1 illustrates the use of `MaxDivSelec` on a 2D synthetic dataset consisting of two gaussians with 40 data instances each. The first gaussian is generated with μ=0 and σ=0.5, while the second has a μ=-3 and σ=1. In the example five data instances are removed from each gaussian.

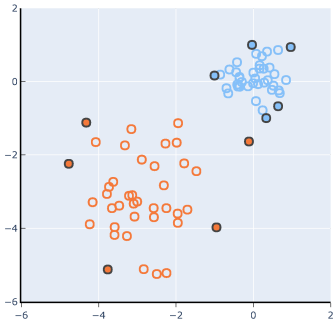


Figure 4.1 Illustration of the data instances selected by `MaxDivSelec` for $\overline{p}_1 = 5$ and $\overline{p}_2 = 5$, which corresponds to 12.5% of the whole dataset.

4.4 Experimental methodology

4.4.1 K-nearest neighbors

In order to prove our hypothesis about effectiveness of `MaxDivSelec` as a data instance selection method for classification models, we had to choose one representative ML model from which our conclusions could be better generalized.

The K-nearest neighbor (KNN) model is a simple yet effective supervised classifier [16]. It predicts the class of an unseen instance by finding its K closest data instances from the training set. The unlabelled instance is then assigned to the majority class among them. The KNN classification model was a natural choice for our experiments for three reasons:

(i) it relies on a distance metric – as well as the MDP.

(iii) it is quite tolerant to outliers and noisy data.

(iii) its classification performance, memory usage and computing times are tightly linked to the number of data instances used for training. [1]

4.4.2 Baseline methods

We compared MaxDivSelec against two other selection methods. The first method, called `Random`, corresponds to our null hypothesis. It simply chooses p data instances to remove at random, with equal probability.

The second method, denoted here `Mahalanobis`, is well-known in the literature [63]. It removes outliers by computing the Mahalanobis distance (4.2) from each data instance to the centroid of the class it belongs. Larger values of the Mahalanobis distance indicate a greater outlier likelihood. The method aims to improve model classification by removing from the training dataset the instances which are too different to be statistically part of a class.

Algorithm 4 presents the pseudo-code of method `Mahalanobis` which returns a set T of data instances for model training. The method computes for each available labelled data instance its Mahalanobis distance to the centroid of the class to which it belongs. In the sequel, the algorithm removes, from each class c, $\overline{p}_c$ data instances whose Mahalanobis distances are the largest computed.

[1] KNN makes use of the so-called *lazy training* or *instance-based learning*. It simply queries over the data to make a prediction [62].

Algorithm 4: `Mahalanobis method`

Input: X: labelled dataset of dimension $n \times s$,
$\overline{p}$: array of dimension $1 \times k$ with the number of instances to be removed per class

$T \leftarrow X$
for $c = 1, \ldots, k$ **do**
  Let $X_c \subseteq X$ be the matrix of dimension $n_c \times s$ composed by the data instances of class c in X
  Compute the covariance matrix Σ_c of X_c
  for each data instance x_ℓ of *class* c **do**
    Compute the Mahalanobis distance $d_\ell = \sqrt{(x_\ell - \mu_c)^{\mathrm{T}} \Sigma_c^{-1} (x_\ell - \mu_c)}$ between x_ℓ and the centroid μ_c of class c
  end
  $R \leftarrow$ the $\overline{p}_c$ instances of class c with largest d
  $T \leftarrow T \setminus R$
end
return T

Figure 4.2 illustrates the application of the `Mahalanobis` method over the same synthetic example of the two the last section. Here, again, five data instances are removed from each gaussian. We remark that the selections performed by `MaxDivSelec` and `Mahalanobis` differ of two data instances only.

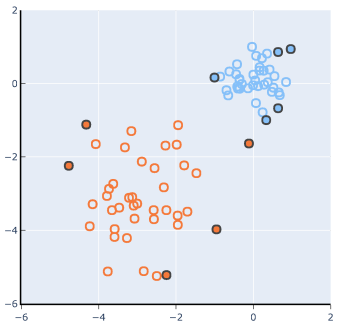


Figure 4.2 Illustration of the data instances selected by `Mahalanobis` for $\overline{p}_1 = 5$ and $\overline{p}_2 = 5$, which corresponds to 12.5% of the whole dataset.

4.4.3 The α parameter

The α parameter controls the percentage of data instances to be removed from the training set. For example, for $\alpha = 50\%$ and $n = 100$, 50 data instances are suppressed from the

training dataset ($p = 50\% \times 100$). This amount is split proportionally across the provided classes. Thus, the method removes $\overline{p}_c = \alpha \times n_c$ data instances from each class $c = 1, \ldots, k$, rounding it to the closest integer. At the end, some adjustments must be performed so that $\sum_{c=1}^{k} \overline{p}_c = p$. Considering $p' = p - \sum_{c=1}^{k} \overline{p}_c$ as the number of adjustments, we either remove or add a data instance to the final training set depending whether p' is positive or negative. The adjustments performed in each class are limited to one, and are performed from the class with the largest amount of data instances to the least populated class. To illustrate, for a training set with 5 classes such that $n_1 = 10$, $n_2 = 10$, $n_3 = 10$, $n_4 = 30$, $n_5 = 40$, for $\alpha = 50\%$ (that is, $p = 50$), we obtain $\overline{p}_1 = 5$, $\overline{p}_2 = 5$,$\overline{p}_3 = 5$, $\overline{p}_4 = 15$, $\overline{p}_5 = 20$. Because $p' = 0$, there is no need to adjust the values. For the same training set, by taking $\alpha = 12.5\%$ (that is, $p = 13$), we have $\overline{p}_1 = 1$, $\overline{p}_2 = 1$, $\overline{p}_3 = 1$, $\overline{p}_4 = 4$, $\overline{p}_5 = 5$. Since, in this case, $p' = 13 - 12 = 1$, $\overline{p}_5$ is adjusted to 6, making a total of $p = 13$ data instances.

4.5 Computational experiments

4.5.1 Datasets

We compare the presented methods over different real-world benchmark datasets. The used datasets are shown in Table 4.1. The different number of classes and attributes across them are aimed to test how well the compared methods handle complex classification problems. All datasets were numerically normalized in each attribute dimension before use.

Table 4.1 Table with datasets' characteristics.

Dataset	n	#classes	#attributes
Iris [38]	150	3	4
Seeds [39]	210	3	7
Dermatology	358	6	34
Ionosphere [41]	351	2	34
Breast cancer Wisc. [42]	683	2	9
Mammographic [43]	830	2	5
Contraceptive [44]	1473	3	9
Abalone [45]	4177	29	8

4.5.2 Evaluation

We used different classification performance metrics depending on whether the classification problem was: (i) binary or multiclass, and (ii) balanced or unbalanced. A binary classification problem is one in which prediction is done for two classes only, while a multiclass problem

involves more than two classes. A balanced problem supposes that the number of data instances of each class is approximately the same, while an unbalanced classification task has the majority of the data instances belonging to a subset of the provided classes. To accommodate those different categories of problems, three performances metrics were used, namely accuracy, RMSE and the F1-score.

The accuracy score is a classification performance metric often used for supervised classification problems [36]. It compares the prediction to the ground-truth class thus computing the ratio of right predictions. In a binary classification problem, there exist four possible cases for a given prediction: a True Positive (TP), a False Positive (FP), a True Negative (TN) and a False Negative (FN) [35]. The two *true* cases happen when the model predict the correct class (positive or negative). Conversely, the *false* cases happen when the model predicts the opposite class. For example, a FP occurs when the model predicts a positive class for an actual negative data instance. The accuracy score is is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

An accuracy value of 1 means that the model predicted 100% of the classes correctly while a score of 0 means that none of the predictions was correct. The accuracy metric can also be generalized for k classes as

$$Accuracy = \frac{\sum_{c=1}^{k} T_c}{\sum_{c=1}^{k} T_c + \sum_{c=1}^{k} F_c} \tag{4.4}$$

where T_c is the number of TP for class c, and F_i the number of FP for that same class.

The Root Mean Squared Error (RMSE) score is a performance metric commonly used for multiclass models [37] for which the classes are ordered somehow. Thus, for an expected value of 0, predicting 1 is less "wrong" than predicting 10, for instance. The RMSE is equal to the squared root of the mean of squared errors between the predictions and the ground-truth values. The formula with y' and y as the predicted and ground-truth values, respectively, and n as the number of predictions is given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2} \tag{4.5}$$

The score represent by how much the model is off on average from the expected values. A score of 0 means that no error was made and the classifier is perfect.

The last and more complex performance metric is the F1-score also called the F-measure.

It is used for unbalanced binary classification problems. It is a suitable score for when the model has to predict well one class in particular amongst others. The *recall* refers to the model's capacity to detect the positive class of interest among the total amount of positive samples, while the *precision* is the model's capacity to well classify TP data instances over the total amount of instances predicted as members of the positive class [35].

The recall and the precision of a model are computed as:

$$recall = \frac{TP}{TP + FN} \qquad precision = \frac{TP}{TP + FP}. \tag{4.6}$$

The F1-score is finally calculated as the geometric mean of both measures:

$$\text{F1-score} = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{4.7}$$

The Iris, Seeds, Dermatology and Contraceptive datasets are assessed according to the accuracy score since they are balanced. The Abalone dataset is an unbalanced dataset with more than two ordered classes. Consequently, KNN's classification performance is evaluated according to the RMSE score for that dataset. Finally, datasets Ionosphere, Breast cancer Wisconsin and Mammographic datasets are evaluated according to F1-score since they consist of binary labelled data, with one majority class.

4.5.3 Cross-validation

The three methods `MaxDivSelec`, `Random` and `Mahalanobis` are tested with the KNN classifier for $K \in \{3, 5, 10\}$ and $\alpha \in \{0.125, 0.25, 0.5\}$, which yields a total of 9 combinations of parameters to be tested. The KNN classifier uses the Euclidean distance. To generalize our results, a 5-fold cross-validation is used to produce multiple test sets. A 5-fold cross-validation separates the data into 5 sets where each set is used as the test set while the rest is used as the training set [12]. That means that 80% of the dataset is used for training while the other 20% is used for the testing. For this experiment, ten 5-fold cross-validation processes are made to produce a total of 50 pairs of training/test sets. The instance selection methods are employed on the training set of each fold. Since `Random` is a stochastic method, it is executed 20 times with a different seed (0 to 19) for each fold.

Table 4.2 reports the benchmark performance scores of the KNN classifier for each dataset. They correspond to the classifier's mean performance when using the whole set of labelled data instances for training, i.e., without instance selection. The datasets are grouped in the table according to the used performance metric.

Table 4.2 Mean benchmark performance of KNN [Accuracy, F1-score, RMSE] for each tested dataset

Dataset	k=3	k=5	k=10
Iris	0.961	0.961	0.964
Seed	0.920	0.930	0.922
Dermatology	0.954	0.956	0.959
Contraceptive	0.458	0.483	0.502
Ionosphere	0.716	0.722	0.707
Breast cancer Wisconsin	0.946	0.953	0.953
Mammographic	0.771	0.789	0.792
Abalone	2.856	2.801	2.692

4.6 Results and Discussion

Our computational results for methods `Random`, `Mahalanobis` and `MaxDivSec` are displayed as box plots to focus on the classification performance distributions of the methods over the tested folds. Besides, we present line charts of the mean performance obtained by each method. We present here a subset of the results, but all box plots can be checked at `https://ktton.github.io/master-research/`[2]. Results are grouped by the number of used neighbors K and by the resulting training size after instance selection. By grouping the results, we can better evaluate how the parameters K and α affect classification performance.

The methods are compared regarding their general behavior, but also on their worst-case result. The worst-case result is the lowest performance result achieved by the method for a given data fold. Regarding accuracy and F1-score that corresponds to the lowest obtained score for a tested fold, whereas for the RMSE tha corresponds to the highest obtained score. The results are further analysed by means of a Wilcoxon statistical test with a confidence level of 5 %. That test tell us if the results achieved by our method are statistically different from those obtained by the baseline methods `Random` and `Mahalanobis`.

4.6.1 Classification performance results

First, we checked the general performance of the instance selection methods for each dataset. For the smallest datasets (regarding n), the three methods presented similar performance. To illustrate that, Figures 4.3a and 4.4a show the results for the dataset Seeds. Moreover, the obtained means are not far from the benchmark KNN performance, which means that using instance selection methods for small datasets does not incur significant losses of classification

[2]There are also presented in the Appendix A.

performance.

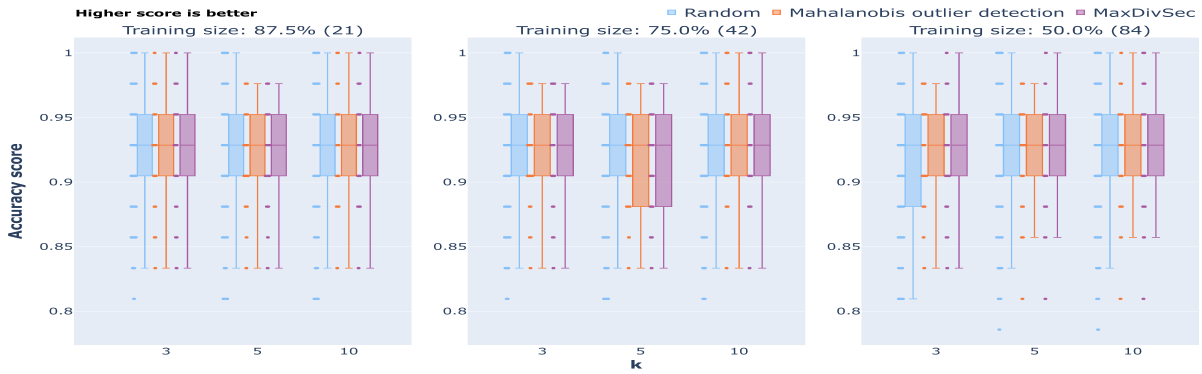
Regarding the largest datasets Breast cancer, Mammographic masses, Ionosphere, Contraceptive and Abalone, we observe a major difference between the classification metrics obtained by the different methods. Performing instance selection with `Random` appear to incur more varied classification performance than by using `Mahalanobis` and `MaxDivSec`, as shown in the box plots of Figures 4.3b, 4.3c and 4.3d. Figures 4.4b, 4.4c and 4.4d show the mean performance of each method for the same three datasets.

The `Mahalanobis` and `MaxDivSec` methods outperform the `Random` method particularly for Ionosphere (Figure 4.4c) and Abalone (Figure 4.4d). Besides, we note across the plots that the `Mahalanobis` and `MaxDivSelec` methods obtain better mean classification scores than those obtained by `Random`. These score differences become larger as more instances are removed from the training sample. In fact, for these instances, the `Mahalanobis` and `MaxDivSelec` methods perform better than the benchmark performance obtained by the KNN classifier using the whole data for training. We can hence conclude that for these datasets restricting the training data to relevant instances is important for increasing the generalization capability of the model.

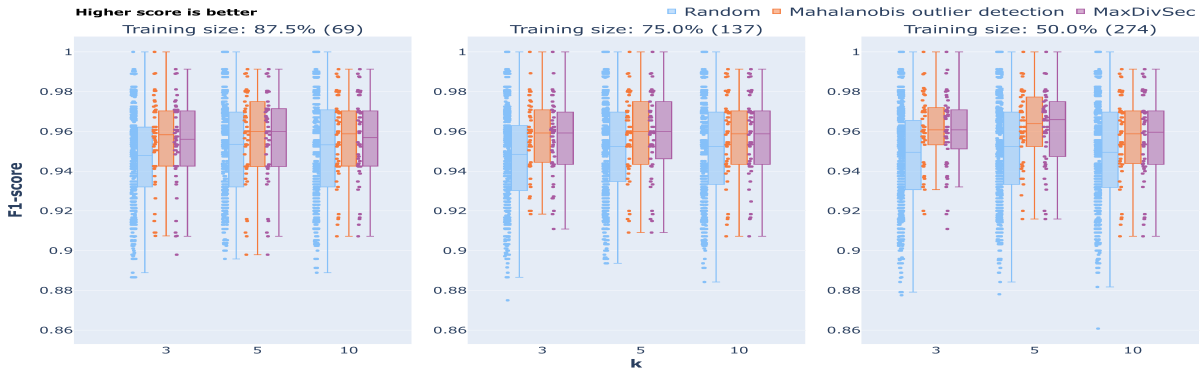
Regarding the worst-case performance, `MaxDivSelec` always obtains better or equal worst-case classification results than `Random`. Having a better worst-case scenario means that our instance selection method is more robust regarding the posterior classification performance of the classifier when predicting the labels of unseen data. When compared to `Mahalanobis`, our method appears to have similar worst-case performance.

Finally, we also analyse the classification performance for varied values of α, and number of neighbors K used by the KNN classifier. Our conclusions are as follows:

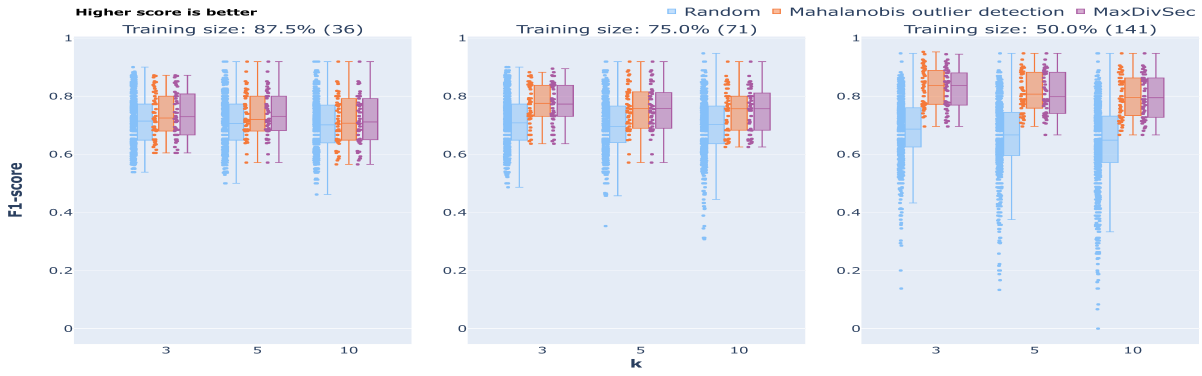
- For the smallest datasets and breast cancer Wisconsin, the classification performance is lightly affected by K and α.
- For both the Abalone and Contraceptive datasets, the classification performance improves as K increases for methods `MaxDivSelec`, `Mahalanobis` and `Random`.
- For the Ionosphere dataset, the classification performance decreases as K increases for methods `MaxDivSelec`, `Mahalanobis` and `Random`, especially with α=50%.
- For the Abalone , Contraceptive, Mammographic and Ionosphere datasets, the classification performance of `MaxDivSelec` and `Mahalanobis` increases as α gets larger, i.e., as more data instances are removed from the training set. They are actually better



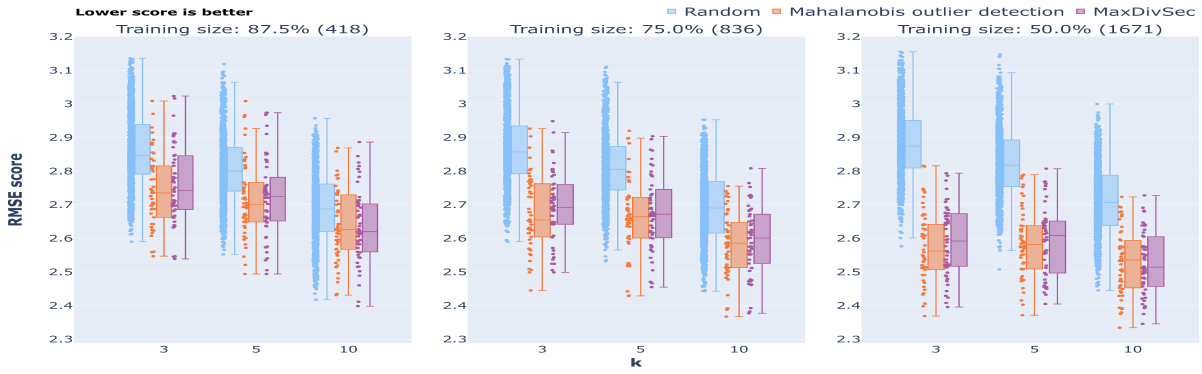
(a) Seeds dataset



(b) Cancer dataset

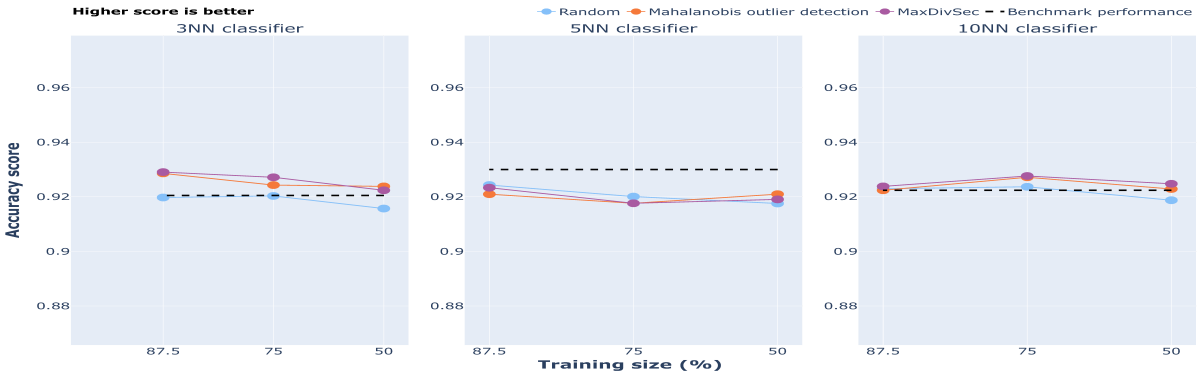


(c) Ionosphere dataset

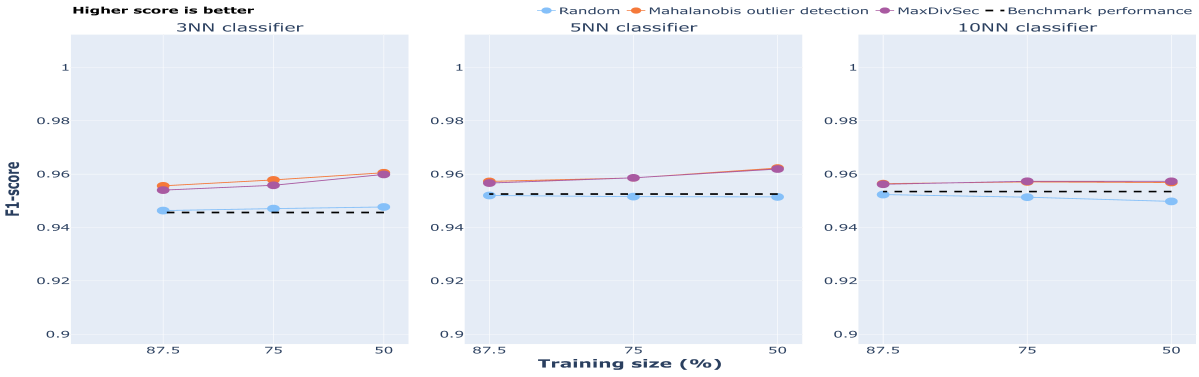


(d) Abalone dataset

Figure 4.3 Boxplot results grouped by training size

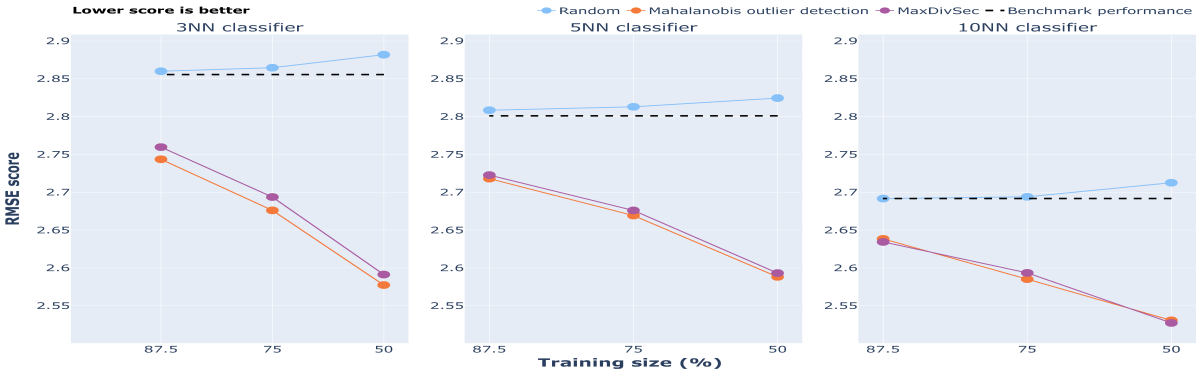


(a) Seeds dataset



(b) Cancer dataset

(c) Ionosphere dataset



(d) Abalone dataset

Figure 4.4 Mean classification results grouped by K

than the benchmark performance presented in Table 4.2 except for the Mammographic dataset.

4.6.2 Wilcoxon tests

This section presents Wilcoxon signed-ranks tests [64] in order to compare the obtained results in terms of statistical significance.

For each dataset, we compared the different methods `Random`, `Mahalanobis` and `MaxDivSelec` on each combination of $K = 3, 5$ and 10, and $\alpha = 0.25, 0.50$ and 0.75, totalizing nine Wilcoxon tests per dataset. Two hypothesis are tested. First, we check if the two result distributions are similar (i.e., the median of differences = 0). If that first hypothesis is rejected, this means that the second hypothesis is true, i.e., that the methods obtain statistically different results (the median of differences <0) We used a confidence level of 5% meaning that the p-value must be smaller than 0.05 to reject an hypothesis. The Wilcoxon test results are reported in Tables 4.3 and 4.4 for each dataset.

Table 4.3 shows the results of the comparisons of the methods `MaxDivSec` and `Mahalanobis` with the method `Random`. We observe that our instance selection method `MaxDivSelec` is statistically different from the `Random` method for most of K and α combinations for all the datasets. We can also verify the same behaviour with the `Mahalanobis` method except for the Seed dataset. The `Mahalanobis` method is only different for two combinations. Moreover, when our method is different from the `Random` method, it is most of the time better.

In Table 4.4, we show the Wilcoxon results obtained when comparing our method `MaxDivSec` with the `Mahalanobis` method. We notice that `MaxDivSec` is seldom different or better than `Mahalanobis` except for two datasets: Contraceptive and Mammographic. However, such difference does not mean that the first is necessarily better than the later. In most of the cases, `MaxDivSelec` is not statistically different from the `Mahalanobis` method.

4.7 Concluding remarks

This paper proposed to investigate the use of diversity for selecting data instances for model training. With that purpose, we proposed `MaxDivSec`, an algorithm that proceeds by removing from the training set of a machine learning model the subset of data instances for which

Table 4.3 Wilcoxon test results with a confidence level of 5% for comparing `MaxDivSec` and `Mahalanobis` with `Random`

Dataset	`MaxDivSec`		`Mahalanobis`	
	Different	Better	Different	Better
Iris	5/9	4/5	7/9	5/7
Seed	4/9	4/4	2/9	2/2
Dermatology	5/9	5/5	6/9	6/6
Ionosphere	9/9	9/9	9/9	9/9
Cancer	8/9	8/8	8/9	8/8
Mammographic	8/9	4/8	6/9	4/6
Contraceptive	9/9	9/9	8/9	8/8
Abalone	9/9	9/9	9/9	9/9

Table 4.4 Wilcoxon test results with a confidence level of 5% for comparing `MaxDivSec` with `Mahalanobis`

Dataset	Different	Better
Iris	1/9	1/1
Seed	0/9	0/0
Dermatology	0/9	0/0
Ionosphere	0/9	0/0
Cancer	1/9	0/1
Mammographic	7/9	3/7
Contraceptive	8/9	8/8
Abalone	3/9	0/3

its associated diversity is maximum. We compared `MaxDivSec`, regarding the classification performance of a target classifier, with two other baseline instance selection methods, one that random selects data instances for suppression and another based on the removal of data outliers. Our results demonstrated that diversity is actually a good criterion for data instance selection as the obtained results by `MaxDivSec` led to superior classification performance in the vast majority of the tested scenarios when compared to the random approach. However, the proposed method was not shown to be significantly different from the method based on the suppresion of outliers.

Finally, although we demonstrated by our experiments that maximum diversity is effective on selecting data instances for model training, its computation still requires the solution of a NP-hard problem either exactly or heuristically. In contrast, the `Mahalanobis` method is genuinely an outlier score for each data point with respect to the class it belongs. As shown by Figures 4.1 and 4.2, outliers are very likely part of maximum diversity subsets because of

their dissimilarities with regard to the other data instances of their classes. However, outliers are generally much faster to compute, and consequently, their removal is indeed recommended for instance selection.

Acknowledgments

The authors would like to thank professors Eduardo Pardo and Abraham Duarte for providing us the OBMA code and executable. This research was partially funded by Natural Sciences and Engineering Research Council of Canada (NSERC) under grants DG-2017–05617 and DG-2020–06311 for its financial support.

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Data availability

The datasets generated during and/or analysed during the current study are available at `https://github.com/ktton/MaxDivSec`.

CHAPTER 5 GENERAL DISCUSSION

In this chapter, we will provide a general discussion about the contributions and results of the present work. In this work, we explored the idea of using diversity as a criterion for an instance selection method with research objective. One was to propose a new instance selection with diversity as the instance selection criterion. The other one was to prove our hypotheses:

- R1: Is diversity a valid criterion for instance selection?
- R2: Is diversity better than other instance selection criteria used in the literature?

5.1 `MaxDivSec`

To demonstrate our research hypotheses, a new instance selection method was developed, namely `MaxDivSec`. It is a decremental hybrid filter method based on the optimization of the maximum diversity problem. Being a decremental method, the algorithm starts with the entire class before using the criterion to choose which instances to remove. It is a hybrid method because the most diverse subset can be composed of both border and interior instances. The maximum diverse subset is composed of instances far/dissimilar from each other. Removing that subset means that we remove the instances that are least likely to be classed together due to their diversity. Thus, the kept data instances are more homogeneous. Unlike when the decision is made instance by instance, the order in which the instances are considered does not matter and does not change the selection. Besides, since the diversity criterion is based on a propriety of the selected subset, we have a more entire view of the selected instances' impact in the class. In our experiments, `MaxDivSelec` is compared against two baselines methods: a random selection and the Mahalanobis outlier detection method.

5.1.1 Performance results

The average performance of the three tested methods was measured for all the considered datasets. After analyzing the results, we observed that our method was better than random data selection. That means that using the max-diversity criterion is worth exploring and produces interesting results. Our first hypothesis, R1, was proven with those results. Diversity is a better criterion than a random selection. However, it had similar statistical performance when compared to the Mahalanobis outlier detection method. Our second hypothesis, R2,

was refuted with those results. Diversity as a criterion was not demonstrated to outperform the compared existing methods.

Intuitively, the Mahalanobis outlier detection makes sense since it gives an outlier score to each instance of a class. Outliers should always be removed from the training set. `MaxDivSec` finds the most diverse subset of instances and removes them. As mentioned previously, we are making the training set more homogeneous to help the model. While this may be beneficial, it can also mislead the model since diversity could be an inherent characteristic of the class. This is an issue that merits further investigation.

5.1.2 Computing times

The Table 5.1 shows the time needed for the model to train with a full training set and a 50% reduced set. The third column contains how much we reduced the training time with the new training set. For all the datasets except Iris and Seed, we achieve a nice reduction %. Those two datasets are the smallest and already need a small amount of time for the training. The other datasets reduce their training time by 46%-65%. However, the pre-processing phase is longer due the the usage of instance selection methods. The ML pipeline end up taking more time. The Figure 5.1, Figure 5.2 and Figure 5.3 compare the time spent on training the model with the full training set with the time spent on selecting a 50% reduced training set and using it to train the model. Only the Random method has a computation time similar to the original pipeline. We see that the new pipeline takes too much time compared to the old one for Mahalanobis and MaxDivSec. MaxDivSec is the slowest of the 3 methods and that can be understood with the time needed to solve the MDP. MDP is an NP-hard problem and solving it exactly may take a lot of time. Heuristics are often used in those cases to provide a solution close to the optimal one. We used an existing heuristic solver called OBMA for our experiments. Even with that heuristic, the time needed to select data instances is greater than the time needed by the Mahalanobis outlier detection method. This is critical since one does not wish to spend much time with data pre-processing, particularly if it does not lead to a large reduction of the training set. We recommend investigating the use of simpler heuristics than OBMA to increase the usability and efficiency of instance selection methods based on maximum diversity.

Table 5.1 Time comparison in seconds between using the full training set and a 50% reduced training set with KNN

Dataset	Full (100%)	Reduced (50%)	Time saved
Iris	0.000659387	0.000542078	18%
Seed	0.000749572	0.000745091	0.6%
Dermatology	0.003828977	0.001341545	65%
Ionosphere	0.002918983	0.001318207	55%
Cancer	0.002746078	0.001473258	46%
Mammographic	0.002203566	0.001189508	46%
Contraceptive	0.005631849	0.002835617	50%
Abalone	0.011400536	0.004921744	57%

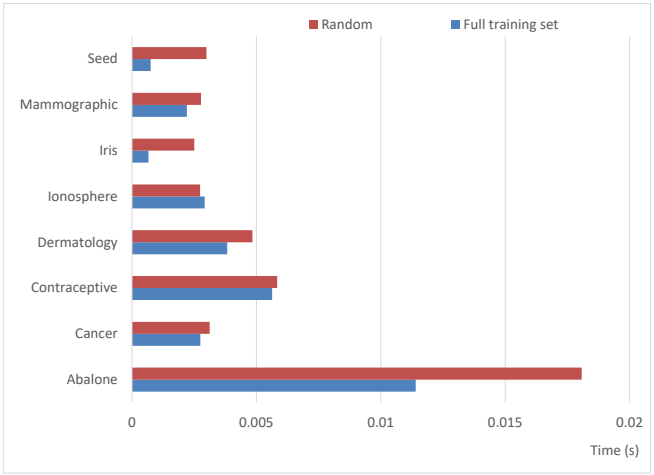


Figure 5.1 Time comparison between using the full training set and using the random selection method.

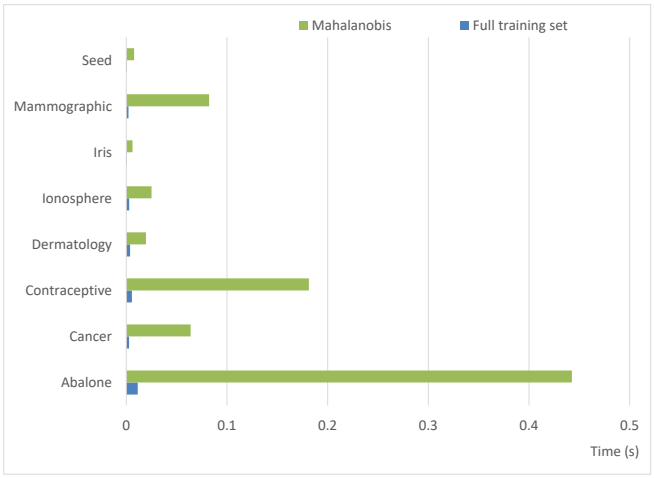


Figure 5.2 Time comparison between using the full training set and using the Mahalanobis selection method.

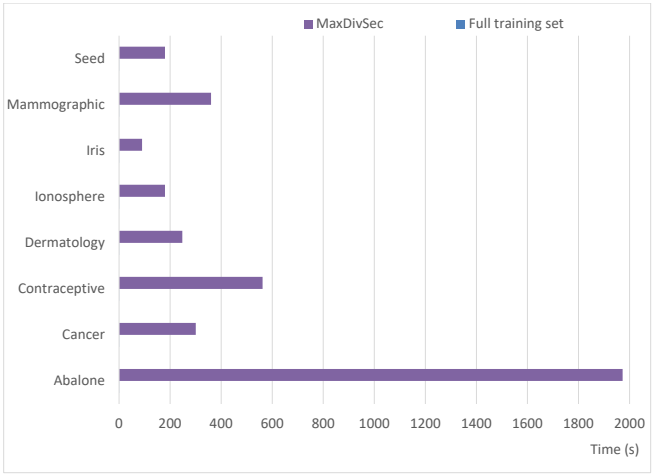


Figure 5.3 Time comparison between using the full training set and using MaxDivSec.

CHAPTER 6 CONCLUSION

6.1 Summary of Works

This thesis proposes a new instance selection method using the MDP as the criterion. This proposition is based on the interest of having a new method to reduce the training set's size for a ML classification model. The concept of diversity seemed to be a good one to apply to remove undesirable instances from a training set.

The thesis started with the basic concepts that are crucial to understanding the proposition and the objectives to be met in Chapter 1. In Chapter 2, we provide a literature review on existing instance selection methods. We wanted to improve on what already existed to create our method. With that information, we chose the maximum diversity as a criterion for our decremental hybrid instance selection method. Chapter 3 presents how the research was conducted and the code used.

The findings were presented in Chapter 4 which presents our results and conclusions that while our method, `MaxDivSec`, is superior to a random approach, it couldn't beat the Mahalanobis outlier detection method for the datasets considered in this research. The performance of `MaxDivSelec` was in fact similar to that of the Mahalanobis-based outlier detection method.

6.2 Limitations

There exist various limitations associated with our work. The first one is the time needed to obtain a solution to the MDP. It is an NP-hard problem and as such, it may take a lot of time to be exactly solved for big datasets.

This brings us to our next limitation: the size of the datasets used for the tests. We couldn't test the method with big datasets. The biggest, Abalone, have around 4000 instances and wouldn't be considered a big dataset in the ML literature. We think that our method wouldn't scale up well if a heavy heuristic is used to obtain MDP solutions.

Another limitation is that we have not optimized our code or used a highly performing machine to run our tests. Our method could be easily run in multiple threads to make it quicker since the selection is done per class. That could help with the total time needed for the data instance selection.

6.3 Future Research

There are several possible venues to improve or iterate on our findings:

- Other diversity criteria can be used for the selection. For example, the p-dispersion is another location problem that was also presented by Kuby [3] that can be explored.
- A combination between the Mahalanobis criterion and the diversity criterion to create a new instance selection method. The Mahalanobis criterion can be used first to remove outliers, before solving the MDP in each class. Another possibility is to compute a combined score obtained from both methods.
- The method can be modified to make it scalable and usable with bigger datasets. Instead of solving the MDP on the entire class, multiple samples could be done to divide the problem into smaller subsets. The instances that are most often chosen are then removed.
- The method can be tested with non-numerical datasets like textual datasets. This gives another context where MaxDivSec could be used with success.
- The method can be used for other ML tasks such as regression or clustering. Clustering is an unsupervised problem and presents a different kind of challenge for MaxDivSec. Regression is sensible to the training set too because the model wants to find a trend among the instances. Both problems use other classifiers/algorithms than KNN.

REFERENCES

[1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," *arXiv e-prints*, p. arXiv:1906.02243, Jun. 2019.

[2] C.-C. Kuo, F. Glover, and K. S. Dhir, "Analyzing and modeling the maximum diversity problem by zero-one programming*," *Decision Sciences*, vol. 24, no. 6, pp. 1171–1185, 1993. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5915.1993.tb00509.x

[3] M. J. Kuby, "Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems," *Geographical Analysis*, vol. 19, no. 4, pp. 315–329, 1987. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.1987.tb00133.x

[4] F. Glover, C.-C. Kuo, and K. S. Dhir, "Heuristic algorithms for the maximum diversity problem," *Journal of Information and Optimization Sciences*, vol. 19, no. 1, pp. 109–132, 1998. [Online]. Available: https://doi.org/10.1080/02522667.1998.10699366

[5] F. Glover, K. Ching-Chung, and K. S. Dhir, "A discrete optimization model for preserving biological diversity," *Applied Mathematical Modelling*, vol. 19, no. 11, pp. 696–701, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0307904X9500083V

[6] A. Duarte and R. Martí, "Tabu search and grasp for the maximum diversity problem," *European Journal of Operational Research*, vol. 178, no. 1, pp. 71–84, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221706000634

[7] K. Ralls *et al.*, "Genetic rescue: A critique of the evidence supports maximizing genetic diversity rather than minimizing the introduction of putatively harmful genetic variation," *Biological Conservation*, vol. 251, p. 108784, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0006320720308429

[8] A. A. Hoffmann, A. D. Miller, and A. R. Weeks, "Genetic mixing for population management: From genetic rescue to provenancing," *Evolutionary Applications*, vol. 14, no. 3, pp. 634–652, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/eva.13154

[9] T. Leinster and M. W. Meckes, "Maximizing diversity in biology and beyond," *Entropy*, vol. 18, no. 3, 2016. [Online]. Available: https://www.mdpi.com/1099-4300/18/3/88

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ch. Introduction, pp. 1–66.

[12] C. Sammut and G. I. Webb, Eds., *Cross-Validation. In: Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, ch. C, pp. 249–249.

[13] ——, *Pruning Set*. Boston, MA: Springer US, 2010, ch. P, pp. 817–817. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_682

[14] J. A. Olvera-López *et al.*, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, may 2010.

[15] S. Garcia *et al.*, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, March 2012.

[16] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[17] P. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory - TIT*, vol. 18, 01 1968.

[18] G. Ritter *et al.*, "An algorithm for a selective nearest neighbor decision rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 21, no. 6, pp. 665–669, 1975.

[19] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, no. 3, pp. 408–421, 1972.

[20] W. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 01 1991.

[21] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, Mar 2000. [Online]. Available: https://doi.org/10.1023/A:1007626913721

[22] R. Paredes and E. Vidal, "Weighting prototypes - a new editing approach," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2, 2000, pp. 25–28 vol.2.

[23] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "Prototype selection via prototype relevance," in *Progress in Pattern Recognition, Image Analysis and Applications*, J. Ruiz-Shulcloper and W. G. Kropatsch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 153–160.

[24] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *J. Artif. Int. Res.*, vol. 6, no. 1, p. 1–34, Jan. 1997.

[25] H. Liu and H. Motoda, "On issues of instance selection," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115–130, Apr 2002. [Online]. Available: https://doi.org/10.1023/A:1014056429969

[26] R. Mollineda, F. Ferri, and E. Vidal, "An efficient prototype merging strategy for the condensed 1-nn rule through class-conditional hierarchical clustering," *Pattern Recognition*, vol. 35, p. 2771–2782, 12 2002.

[27] J. C. Riquelme, J. S. Aguilar-Ruiz, and M. Toro, "Finding representative patterns with ordered projections," *Pattern Recognition*, vol. 36, no. 4, pp. 1009–1018, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132030200119X

[28] G. Shakhnarovich, T. Darrell, and P. Indyk, "Nearest-neighbor methods in learning and vision: Theory and practice (neural information processing)," 2005.

[29] C. Chethana, "Prediction of heart disease using different knn classifier," Piscataway, NJ, USA, 2021//, pp. 1186 – 94, kNN algorithms;optimized KNN;prediction speed;different KNN classifier;Machine learning classification algorithms;cloud storage;websites;heart disease dataset;MATLAB;misclassification rate;distance weight;. [Online]. Available: http://dx.doi.org/10.1109/ICICCS51141.2021.9432178

[30] Z. Liu, L. Wang, and K. Chen, "Secure efficient federated knn for recommendation systems," Cham, Switzerland, 2021//, pp. 1808 – 19, privacy requirements;federated model;global neighbors;high-quality models;cross-domain training;SF-KNN;data sources;precise neighbors;local KNN;secure efficient federated KNN;recommendation systems;querying neighbors;training data;data owner;novel KNN approach;secured federated KNN;. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-70665-4_195

[31] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, no. 1, pp. 46–52, Mar 1985. [Online]. Available: https://doi.org/10.1007/BF02776078

[32] C. Faloutsos and K.-I. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," *SIGMOD Rec.*, vol. 24, no. 2, p. 163–174, May 1995. [Online]. Available: https://doi.org/10.1145/568271.223812

[33] X. Wang *et al.*, "An index structure for data mining and clustering," *Knowledge and Information Systems*, vol. 2, no. 2, pp. 161–184, Jun 2000. [Online]. Available: https://doi.org/10.1007/s101150050009

[34] R. O. Duda *et al.*, *Pattern classification*, 2nd ed. New York ;: John Wiley & Sons, 2001. [Online]. Available: http://catdir.loc.gov/catdir/toc/onix03/99029981.html

[35] K. M. Ting, *Precision and Recall. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning.* Boston, MA: Springer US, 2010, ch. P, pp. 781–781.

[36] C. Sammut and G. I. Webb, Eds., *Accuracy. In: Encyclopedia of Machine Learning.* Boston, MA: Springer US, 2010, ch. A, pp. 9–10.

[37] C. Sammut and G. I. Webb, *Mean Squared Error. In: Encyclopedia of Machine Learningr.* Boston, MA: Springer US, 2010, ch. M, pp. 653–653.

[38] R. Fisher, "UCI machine learning repository: Iris data set," 1936. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris

[39] M. Charytanowicz *et al.*, "UCI machine learning repository: Seeds data set," 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/seeds

[40] G. Demiroz, H. A. Govenir, and N. Ilter, "UCI machine learning repository: Dermatology data set," 1998. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/dermatology

[41] V. G. Sigillito *et al.*, "UCI machine learning repository: Ionosphere data set," 1989. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/ionosphere

[42] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc Natl Acad Sci U S A*, vol. 87, no. 23, pp. 9193–9196, Dec 1990.

[43] M. Elter, R. Schulz-Wendtland, and T. Wittenberg, "UCI machine learning repository: Mammographic mass data set," 2007. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Mammographic+Mass

[44] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "UCI machine learning repository: Contraceptive method data set," 1999. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

[45] W. J. Nash *et al.*, "UCI machine learning repository: Abalone data set," 1994. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Abalone

[46] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006. [Online]. Available: http://jmlr.org/papers/v7/demsar06a.html

[47] Y. Zhou, J.-K. Hao, and B. Duval, "Opposition-based memetic search for the maximum diversity problem," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 731–745, 2017.

[48] D. Pyle, *Data Preparation for Data Mining*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[49] S. Zhang, C. Zhang, and Q. Yang, "Data preparation for data mining," *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 375–381, 2003. [Online]. Available: https://doi.org/10.1080/713827180

[50] G. I. Webb, *Overfitting. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning.* Boston, MA: Springer US, 2010, ch. O, pp. 744–744.

[51] L. C. Molina, L. Belanche, and A. Nebot, "Feature selection algorithms: a survey and experimental evaluation," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, 2002, pp. 306–313.

[52] T. Dasu and T. Johnson, *Exploratory Data Mining and Data Cleaning*, 1st ed. USA: John Wiley & Sons, Inc., 2003.

[53] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: a review," *Artificial Intelligence Review*, vol. 52, no. 2, pp. 857–900, Aug 2019. [Online]. Available: https://doi.org/10.1007/s10462-017-9611-1

[54] M. Blachnik, "Ensembles of instance selection methods: a comparative study," *International Journal of Applied Mathematics and Computer Science*, vol. 29, no. 1, pp. 151 – 68, Mar. 2019, preprocessing methods;feature selection;bagging ensemble;instance selection algorithms;feature bagging ensemble;AdaBoost ensemble;additive noise

ensemble;1NN classifier;kNN classifier;SVM classifier;. [Online]. Available: http://dx.doi.org/10.2478/amcs-2019-0012

[55] A. Zimek and P. Filzmoser, "There and back again: Outlier detection between statistical reasoning and data mining algorithms," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 6, p. e1280, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1280

[56] T. von Ghyczy, "Product diversity and proliferation as a new mode of competing in the motor car," *Int. J. of Vehicle Design*, vol. 6, no. 4/5, pp. 423–425, 1985.

[57] M. Lozano, D. Molina, and C. García-Martínez, "Iterated greedy for the maximum diversity problem," *European Journal of Operational Research*, vol. 214, no. 1, pp. 31–38, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221711003626

[58] R. Martí, M. Gallego, and A. Duarte, "Optsicom project," 2010. [Online]. Available: http://grafo.etsii.urjc.es/optsicom/mdp/

[59] R. Martí, M. Gallego, and A. Duarte, "A branch and bound algorithm for the maximum diversity problem," *European Journal of Operational Research*, vol. 200, no. 1, pp. 36–44, 2010.

[60] R. Martí *et al.*, "Discrete diversity and dispersion maximization: A review and an empirical analysis from an OR perspective," *submitted March 2021.*

[61] G. Strang, *The Singular Value Decomposition (SVD). In: Introduction to Linear Algebra*, 5th ed. Wellesley, MA: Wellesley-Cambridge Press, 2009, ch. 7, pp. 364–400.

[62] E. Keogh, *Instance-Based Learning.* Boston, MA: Springer US, 2010, ch. I, pp. 549–550. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_409

[63] J. Fernández Pierna *et al.*, "Methods for outlier detection in prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 63, no. 1, pp. 27 – 39, 2002, chemometrics 2002 S.I. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743902000345

[64] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://www.jstor.org/stable/3001968

APPENDIX A RESULTS

Here are the figures showing the results obtained for each dataset during the experimentation that couldn't be put in the article. The black dashed line in the figures represents the performance of the classifier when the whole data is used for the training.

Figure A.1 Iris dataset

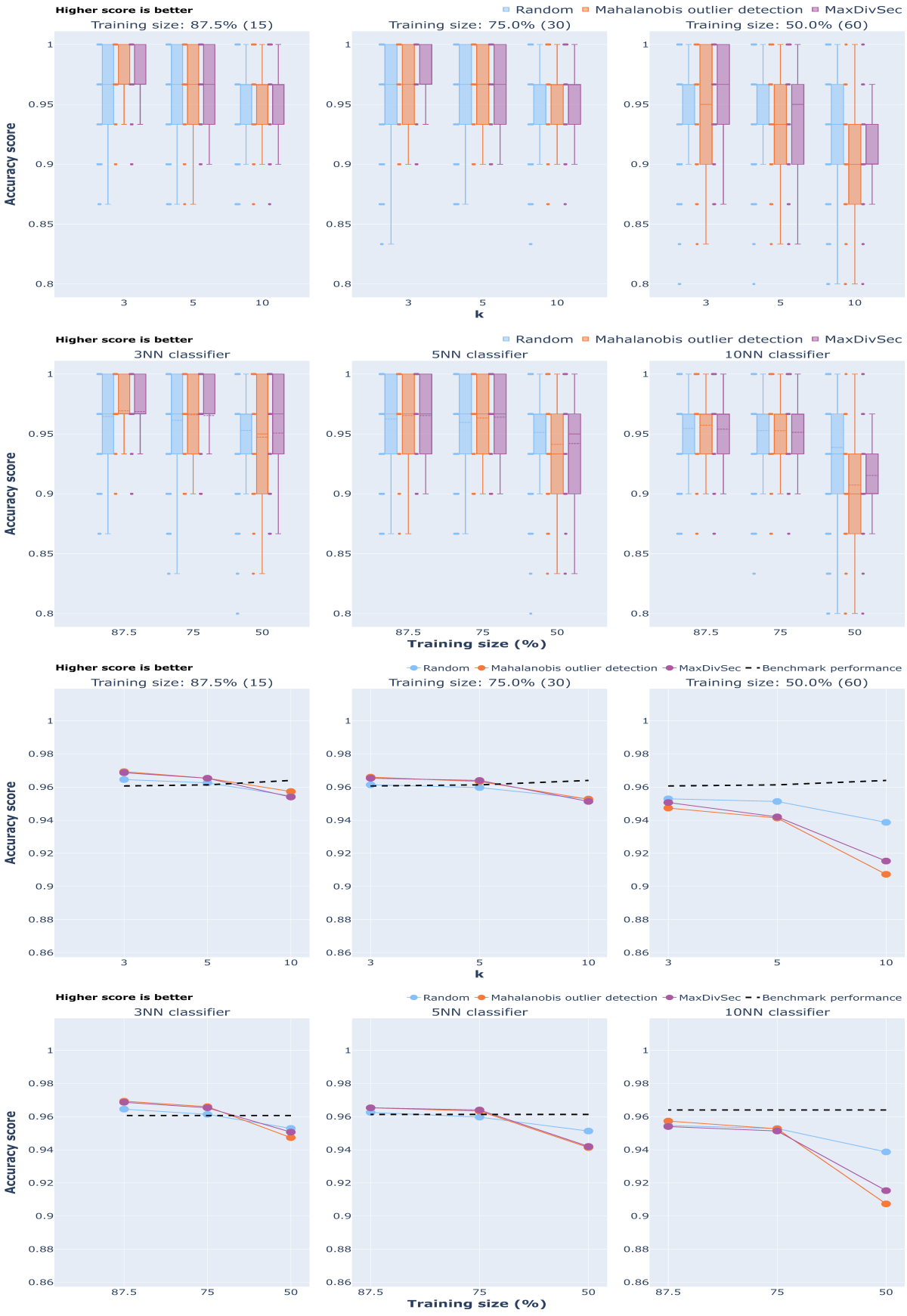


Figure A.2 Seeds dataset

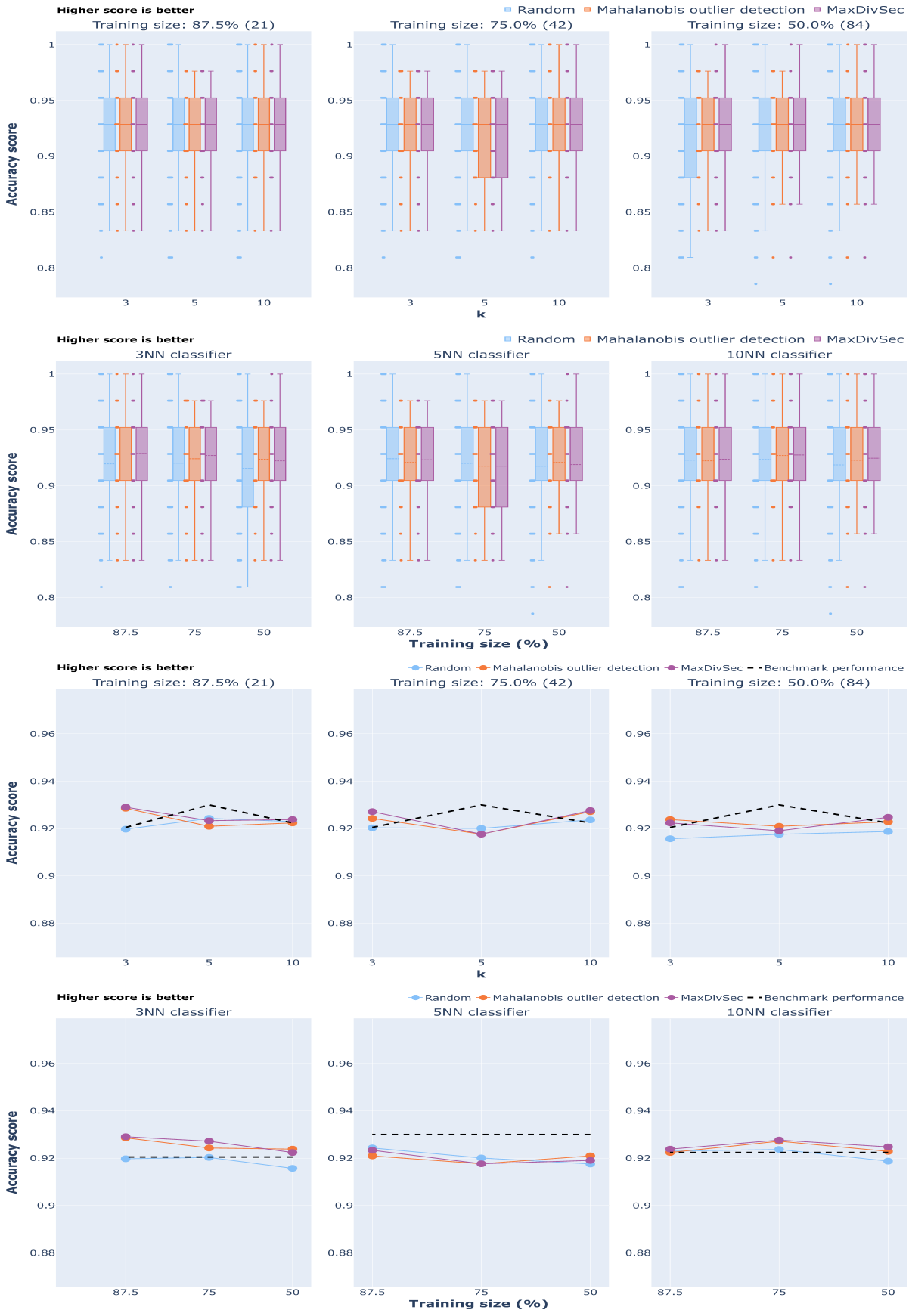


Figure A.3 Dermatology dataset

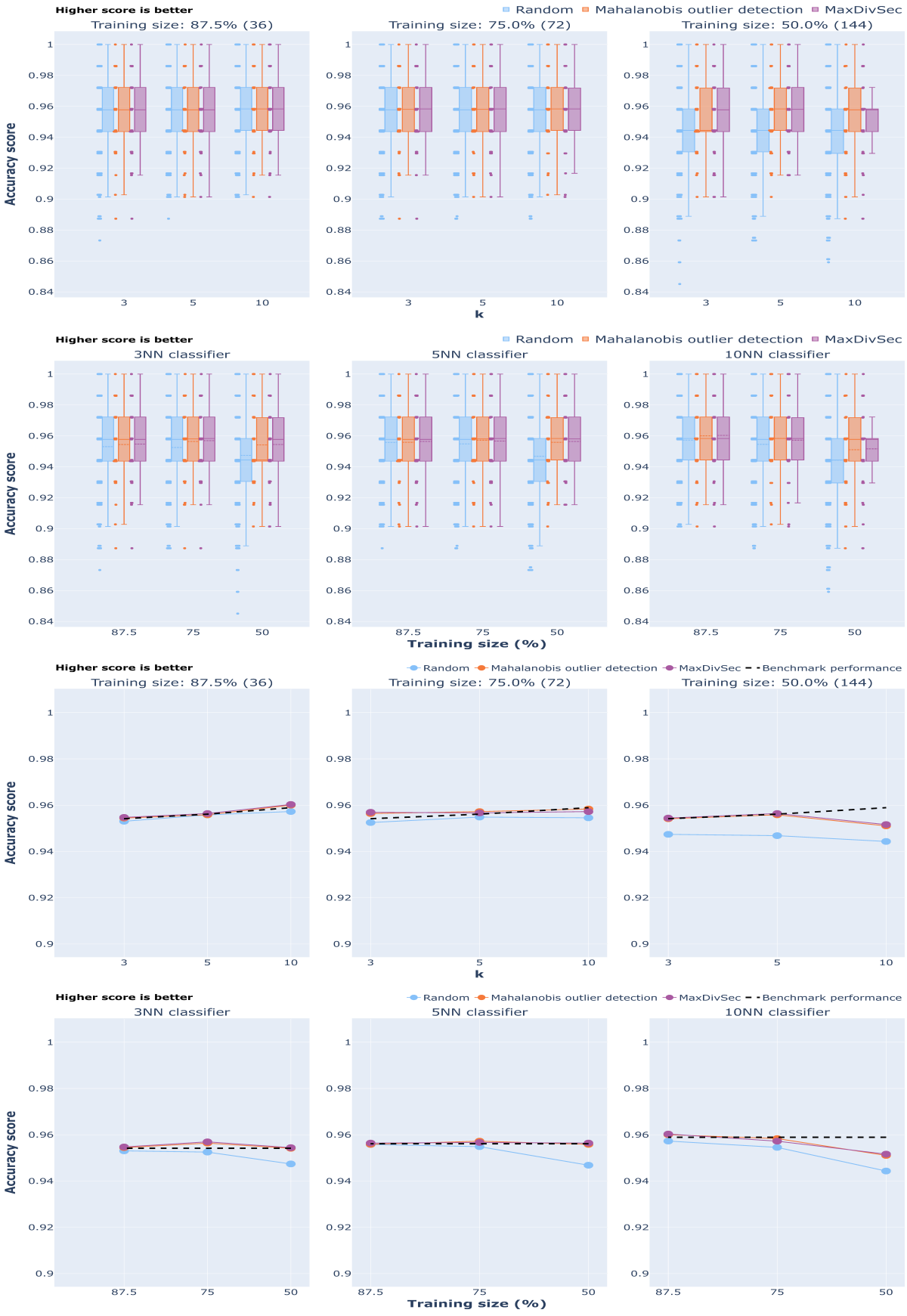


Figure A.4 Ionosphere dataset

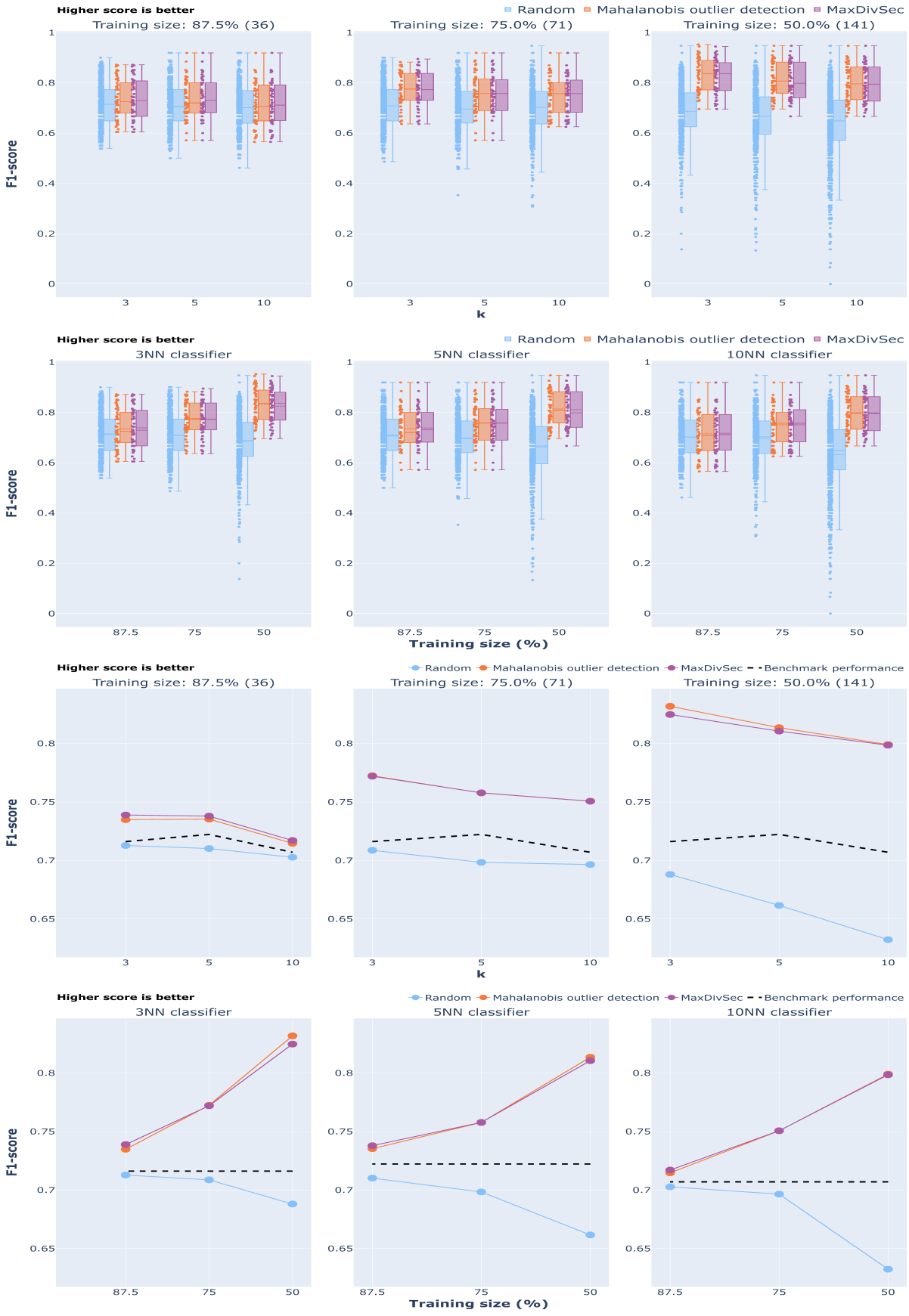


Figure A.5 Breast cancer wisconsin dataset

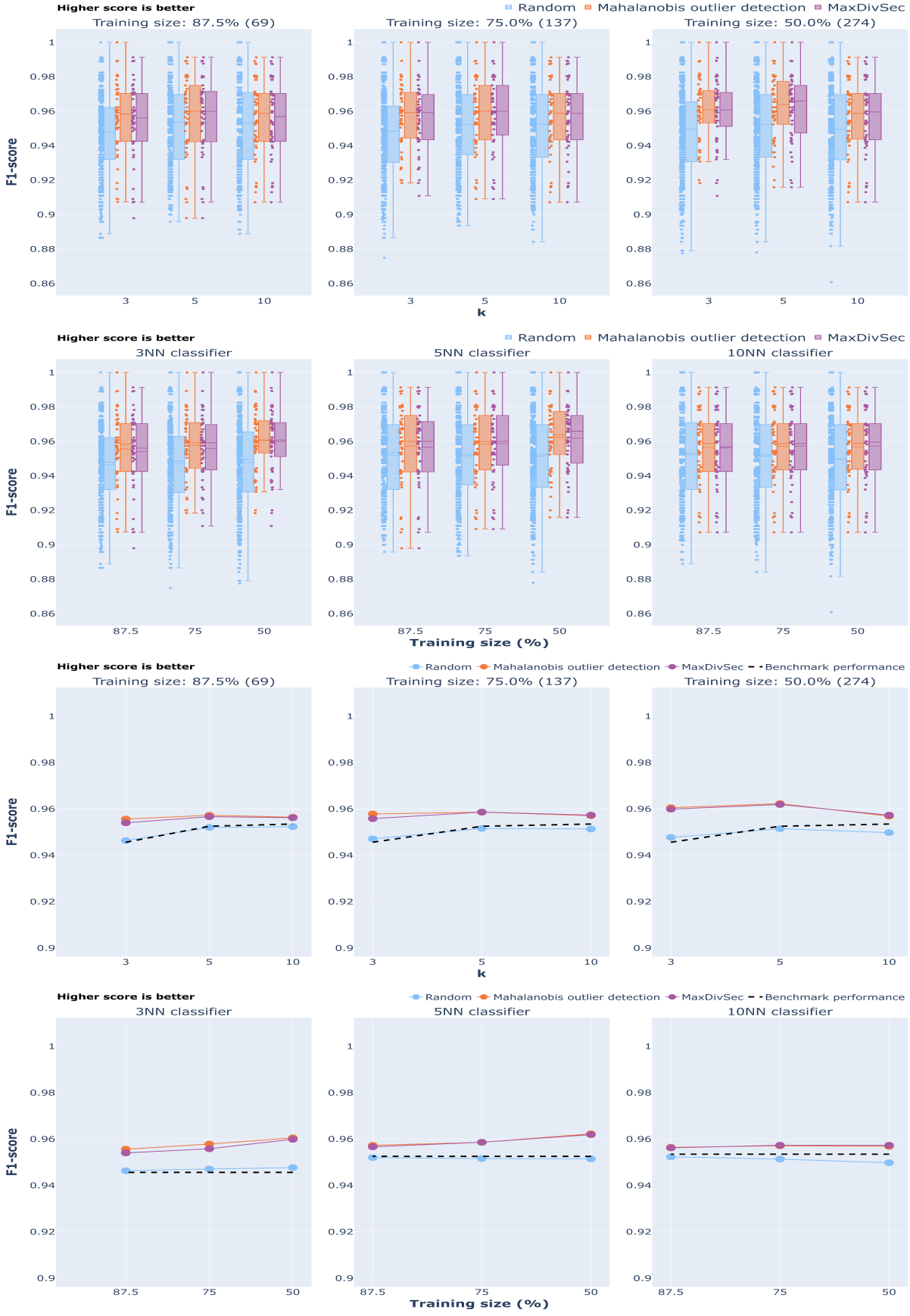


Figure A.6 Mammographic dataset

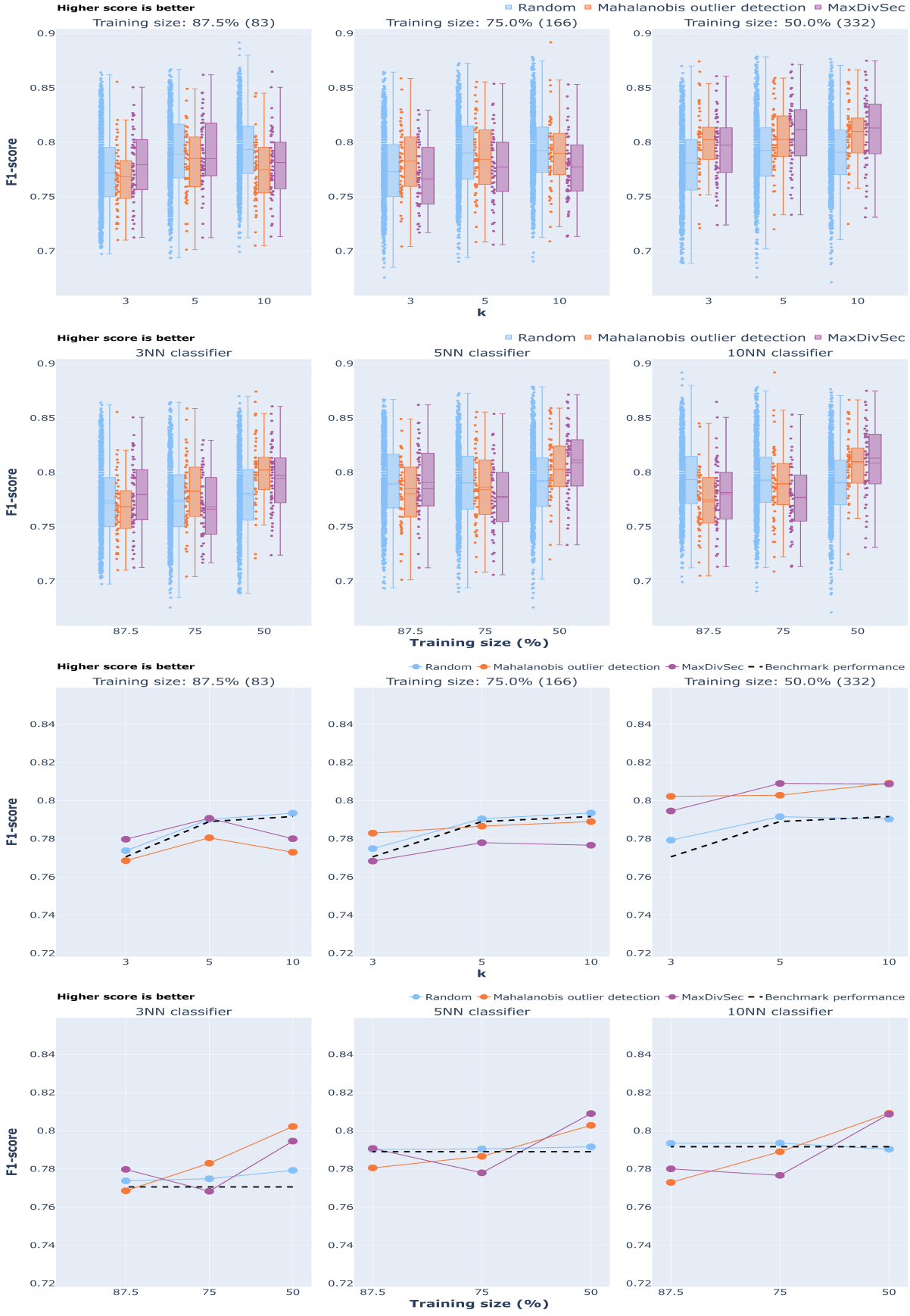


Figure A.7 Contraceptive dataset

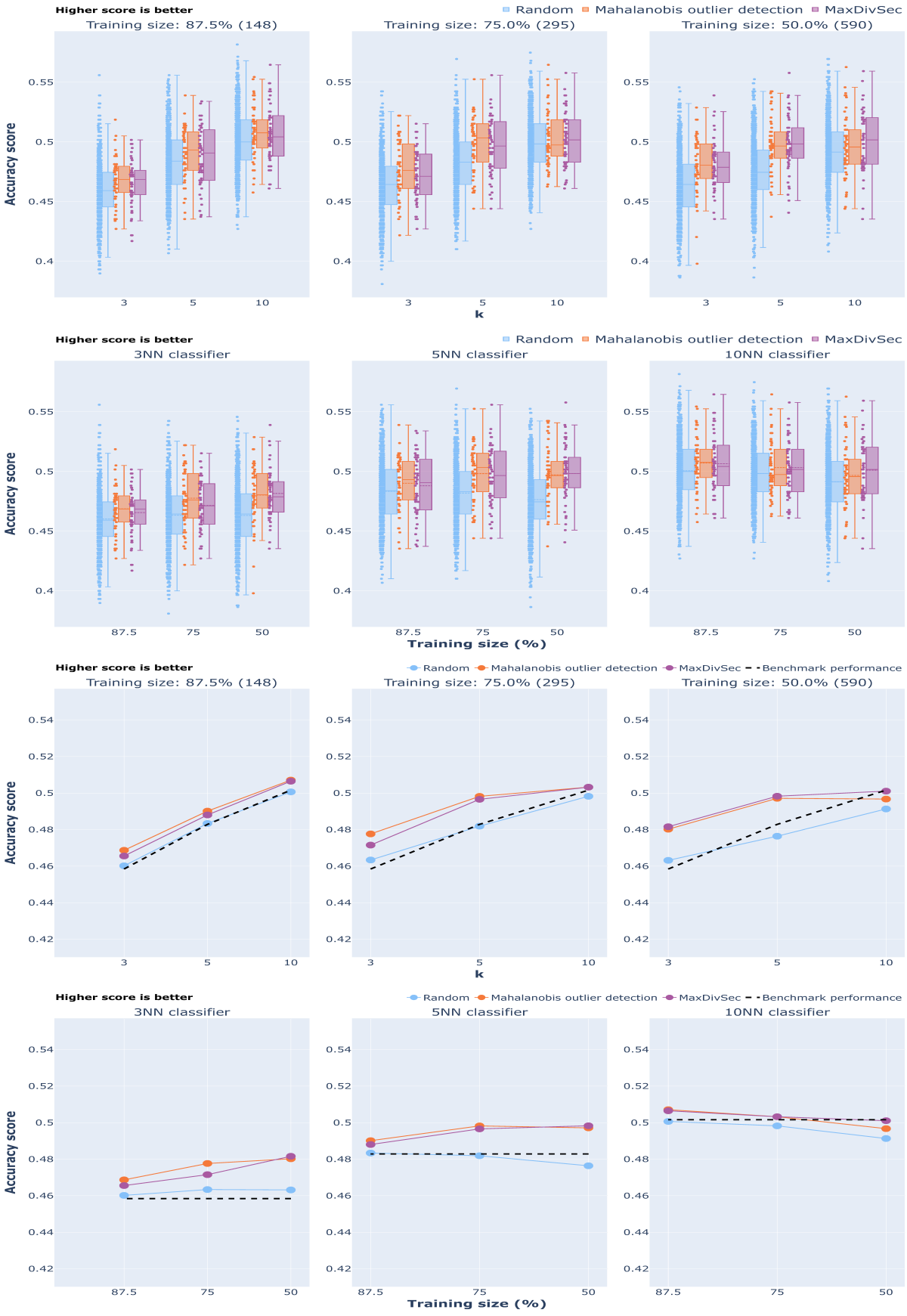


Figure A.8 Abalone dataset