REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

CONSTRUCTION AND PERFORMANCE ANALYSIS OF LOCALLY ADAPTIVE BASE AND ENSEMBLE LEARNERS

FARUK BULUT

PhD. THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM OF COMPUTER ENGINEERING

ADVISER
ASSISTANT PROF. DR. M. FATİH AMASYALI

İSTANBUL, 2015

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

CONSTRUCTION AND PERFORMANCE ANALYSIS OF LOCALLY ADAPTIVE BASE AND ENSEMBLE LEARNERS

A thesis submitted by Faruk BULUT in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY** is approved by the committee on 03.27.2015 in Department of Computer Engineering, Computer Engineering Program.

Thesis Adviser

Assist. Prof. Dr. M. Fatih AMASYALI

Yıldız Technical University

Approved By the Examining Committee

Assist. Prof. Dr. M. Fatih AMASYALI

Yıldız Technical University ______________________

Prof. Dr. Selim AKYOKUŞ, Member

Doğuş University ______________________

Assoc. Prof. Dr. Banu DİRİ, Member

Yıldız Technical University ______________________

Assoc. Prof. Dr. Songül ALBAYRAK, Member

Yıldız Technical University ______________________

Assoc. Prof. Dr. Olcay KURŞUN, Member

İstanbul University ______________________

ACKNOWLEDGEMENTS

I would like to express my deepest appreciations to my adviser Assist. Prof. Mehmet Fatih AMASYALI for his guidance, valuable suggestions and creative comments throughout the research. It would not be possible to complete this thesis without his persistent help and patience. I consider myself very fortunate for being able to work with a very considerate and encouraging adviser like him.

I also want to show my sincere gratitude to the members of the thesis committee, Prof. Dr. A. Coşkun SÖNMEZ and Assoc. Prof. Dr. Olcay KURŞUN. I owe many thanks to them for their valuable comments and suggestions while studying on the thesis.

In addition, I express my thanks and appreciations to my mother, my wife and my children for being understanding and patient. I dedicate this study to them.

March, 2015

Faruk BULUT

TABLE OF CONTENTS

LIST OF SYMBOLS

N	Number of elements (samples, points)
k	Number of nearest points
l	Number of clusters for *k*-means
M	Number of points that will be clustered
$a_i(x)$	the *i*th attribute of the query point *x*
D	Number of dimension
A	Number of attribute
$f(x_i)$	Regression of classification function for x_i
μ_i	Mean value of the *i*th attribute
σ_i	Variance of the *i*th attribute
$\mu(x)$	Classifier function
$\mu_i^c(x)$	A function of the *i*th expert calculating the probability of being in the class *c* for the test point *x*
C	Number of class labels
c_i	*i*th class
w_i	The weight of the *i*th expert
E_j	The medoid of the *i*th expert
$\mathrm{dist}(x, E_j)$	A function finding the distance between x and E_j
g_i	The weight of the *i*th expert
G	Sum of all the experts weights
f	Fisher's discriminant ratio
$R(d)$	Directional vector Maximum fisher's discriminant retio
$\vec{d}$	Directional vector
$\vec{\mu}_i$	Mean vector
B	Class scatter matrix
$\max(f_i, c_j)$	Maximum values of the feature f_i for the class c_j
$\min(f_i, c_j)$	Minimum values of the feature f_i for the class c_j

LIST OF ABBREVIATIONS

k-NN	*k* Nearest Neighbors
1NN	One Nearest Neighbors
NN	Nearest Neighbors
BSP	Binary Space Partitioning
BST	Binary Search Tree
k-D	k Dimensional
EM	Expectation Maximization
SVM	Support Vector Machines
MoE	Mixture of Experts
DT	Decision Tree
1NC	One Nearest Neighbors
LVQ	Learning Vector Quantization
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
Acc	Accuracy
Err	Error
Cfs	Correlation Feature Selection
LOO	Leave One Out
CV	Cross Validation
MCC	Matthews Correlation Coefficient
WEKA	Waikato Environment for Knowledge Analysis
ARFF	Attribute-Relation File format
FA	Factor Analysis
CC	Correlation Coefficient
MLP	Multilayer Perceptron
UCI	University of California, Irvine
ID3	Iterative Dichotomiser 3
LP	Linear Programming
F1	Maximum Fisher's discrimination ratio
F1v	Directional-vector Max. Fisher's discriminant ratio
F2	Volume of overlap region
F3	Maximum (individual) feature efficiency
F4	The collective feature efficiency

L1	Minimized error by Linear Programming (LP)
L2	Error rate of linear classifier by LP
L3	Nonlinearity of classifier by LP
N1	Fraction of points on boundary (Minimum Spanning Tree method)
N2	Ratio of average intra/inter class NN distance
N3	Leave One Out Error rate of 1NN classifier
N4	Nonlinearity of 1NN classifier
T1	Fraction of points with associated adherence subsets retained
T2	Average number of points per dimensions
DCoL	Documentation for the Data Complexity Library in C++
KEEL	Knowledge Extraction (based on) Evolutionary Learning
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
SMO	Sequential Minimal Optimization
MST	Minimum Spanning Tree
CART	Classification And Regression Tree
TB-RBF	Decision Tree based Radial Basis Function
URL	Uniform Resource Locator
SMO	Sequential Minimal Optimization

LIST OF FIGURES

Page

LIST OF TABLES

Page

ABSTRACT

CONSTRUCTION AND PERFORMANCE ANALYSIS OF LOCALLY ADAPTIVE BASE AND ENSEMBLE LEARNERS

Faruk BULUT

Department of Computer Engineering

PhD Thesis

Adviser: Assist. Prof. Dr. M. Fatih AMASYALI

In this study, construction and performance analysis of locally adaptive base and ensemble learners have been proposed by using Meta and Ensemble Learning techniques. The characteristics and meta-features of the discretized sub regions in a dataset have been analyzed for the purpose of better learning performance. A detailed performance analysis of a local base learner over any type of dataset is firstly performed in order to understand the reasons of both failure and success in classification. Additionally, the discrete sub regions are learned by using the Mixture of Experts model to enhance the overall prediction accuracy. Furthermore, a localized lazy base learner using a dynamic parameter creator mechanism is established to gain better performance.

Firstly, prediction of the performance of a local base learner (e.g., decision tree) is proposed by using Meta Learning methods. We have selected some datasets and some extracted geometrical complexity measures from the datasets so as to use in Meta Learning. The extracted features and the real accuracy rates of these classifiers have been accepted as attributes and class labels respectively, and they are placed into the Meta Learning dataset. With this training set, it becomes possible to predict the accuracy of a decision tree on upcoming datasets. Moreover, by using the new meta-learning dataset, a feasible linear regression model has been built for the purpose of predicting the performance of a decision tree classifier. As a consequence, some meaningful reasons have been determined why decision trees outperform or fail on any dataset.

Secondly, a new approach in Mixture of Experts using hard clustering techniques is presented for accurate prediction and classification. Mixture of Experts, as one of the popular ensemble methods developed in recent years, is used to have higher prediction performance in classification and regression problems. In this technique, a dataset is divided into sub regions through a soft clustering procedure. An expert for each region is assigned and trained with the corresponding data points. The decisions of the experts are combined by a gate function in order to predict the class label of a query point. In contrast to the traditional Mixture of Experts method, in this study, a dataset is divided into regions by a hard clustering technique and the class prediction method is performed by four different types of proposed gate functions: cooperating, competitive, commensurative, and Borda count. In the experiments, better performances have been obtained with the proposed cooperating gate function due to its mechanism that gives different weights to the experts in the network.

Finally, a locally adaptive parameter selection mechanism for nearest neighbor classifiers using clustering methods is suggested for more accuracy. The *k* Nearest Neighbors classification technique has a worldwide fame due to its simplicity, effectiveness and robustness. As a lazy learner, *k* Nearest Neighbors used in numerous fields is a versatile algorithm. In this classifier, the *k* parameter is generally chosen by the user and the optimal *k* value is found by experiments. The chosen constant *k* value is used during the whole classification phase. The same *k* value used for each test sample during the validation step might decrease the overall prediction performance. The optimal *k* value for each test data point should vary in order to have more accurate predictions. In this study, a dynamic *k* value selection method for each instance is proposed. This improved classification method employs a simple clustering procedure. In the experiments, more accurate results have been found. The reasons of success have also been understood and presented.

Key words: Meta-complexities, mixture of experts, ensemble and meta learning

ÖZET

VERİ UZAYININ BÖLGESEL ÖZELLİKLERİNİ KULLANAN TEKİL VE KOLEKTİF ÖĞRENİCİ TASARIMLARI VE PERFORMANS ANALİZLERİ

Faruk BULUT

Bilgisayar Mühendisliği Anabilim Dalı

Doktora Tezi

Tez Danışmanı: Yrd. Doç. Dr. M. Fatih AMASYALI

Kolektif ve Meta Öğrenme yöntemlerini temel alan çalışmamızda var olan temel öğrenicilerin yanı sıra, yeni temel öğreniciler de kullanılarak sınıflandırıcılarda performans analizi ve artırımı üzerine teorik ve pratik çalışmalar yapılmıştır. Bir veri setinde bulunan ayrık alt bölümlerin karakteristik ve meta özellikleri daha iyi bir öğrenme başarısı için analiz edilmiştir. Öncelikle bir Karar Ağacı sınıflandırıcısının performans analizi detaylı bir şekilde yapılmış ve sınıflandırma başarısının ya da başarısızlığının nedenleri veri setindeki lokal özelliklere bakılarak araştırılmıştır. Ayrıca, veri setinin ayrık alt bölümleri, Uzman Karışımlarında öne sürülen yeni bir yaklaşım ile ele alınmış ve toplam sınıflandırma başarısı artırılmıştır. Son olarak örnek tabanlı bir öğrenicinin performansı, veri setinin lokal özelliklerine bağlı olarak dinamik parametre seçimi yapan bir mekanizma güçlendirilmiştir.

İlk olarak, Karar Ağaçlarında performans tahmininin Meta Öğrenme yöntemleri yardımı ile yapılması üzerine bir çalışma gerçekleştirilmiştir. Geometrik karmaşıklık ölçütleri, iki sınıflı veri setlerinden elde edilerek Meta Öğrenmede kullanılmıştır. Çıkarılan bu ölçütlerin her biri Meta öğrenme veri setinde öznitelik olarak belirlenmiştir. Ayrıca her bir veri seti üzerindeki karar ağaçlarının elde edilen performansı ise Meta öğrenme setine sınıf etiketi olarak atanmıştır. Bu sayede oluşturulan eğitim seti ile karar ağaçlarının başarısı regresyon teknikleriyle tahmin edilebilmiştir. Ayrıca bu eğitim seti ile performans analizi yapabilen geçerli ve anlamlı bir lineer regresyon modeli çıkarılabilmiştir. Sonuç olarak karar ağaçlarının bir veri seti üzerinde neden başarılı ya da başarısız olduğu anlaşılabilmiştir. Yapılan testlerde tahmin yönteminin az düzeyde hata yaptığı gözlemlenmiştir.

Daha sonra, kolektif öğrenme yöntemlerinden biri olan Uzman Karışımlarında yeni bir yaklaşımın katı kümeleme yöntemiyle sunulması üzerine bir çalışma yapılmıştır. Uzman karışımları, öğrenme ve sınıflandırma başarısını artırmak için kullanılan yöntemlerden biridir. Bu yöntemde veri seti yumuşak kümeleme ile bölümlere ayrılarak her bir bölüm için ayrı bir uzman atanır ve o bölümdeki örneklerle eğitilir. Geçiş fonksiyonu ile de uzmanların kararları birleştirilerek sınıflandırma işlemi yapılır. Herhangi bir sınıflandırıcı uzman olabileceği gibi yüksek performans, hız ve şeffaflıklarından ötürü karar ağaçlarının literatürde tavsiye edildiği görülmektedir. Bu çalışmada ise veri seti, bilinenin aksine yumuşak kümeleme yerine katı kümeleme yöntemiyle alt veri setlerine bölünmüş ve her bir alt veri seti için ayrı bir karar ağacı inşa edilmiştir. Geliştirilen dört farklı geçiş fonksiyonu modeli ile uzmanların kararları birleştirilmiştir. Bunlar işbirlikçi, yarışmacı, orantılı ve Borda sayımıdır. Deneysel çalışmalarda işbirlikçi yöntemin sahip olduğu mekanizmadan ötürü diğerlerine göre daha yüksek başarı gösterdiği gözlemlenmiştir. İşbirlikçi geçiş fonksiyonun tasarlanmasında test noktasına uzakta bulunan uzmanların etkisinin daha az; yakında olanların etkisinin ise daha fazla olması gerektiği düşüncesinden yola çıkılarak Shepard metodundan yararlanılmış ve ortak komite kararı bulunmuştur.

Son olarak örnek tabanlı sınıflandırıcılar için adaptif ve dinamik parametre seçiminin denetimsiz öğrenme teknikleri yardımıyla bulunması üzerine teorik ve pratik bir çalışma yapılmıştır. Örnek tabanlı sınıflandırıcılar basitliği, uygulanabilirliği ve şeffaflığından ötürü yaygın bir kullanıma sahiptir. *k* en yakın komşuluk sınıflandırıcısı bu alanda en çok tercih edilen algoritmalardan biridir. *k* en yakın komşuluk sınıflandırıcısında performans, k parametresi ile doğrudan ilişkilidir. En uygun k parametresi, kullanıcı tarafından genellikle deneme-yanılma yöntemiyle seçilir. Bununla birlikte, bir veri setinde çapraz geçerleme işlemi süresince her bir test örneği için aynı *k* parametresinin kullanılması genel sınıflandırma başarısını olumsuz etkilemektedir. Her bir test örneği için en uygun *k* değerinin seçilmesi daha başarılı sonuçlar elde edilmesini sağlayabilmektedir. Çalışmamızda her bir test örneği için en uygun k parametresini kümeleme yöntemiyle bulan ve bu sayede genel sınıflandırma başarısını artıran bir yöntem üzerinde çalışılmış ve başarılı sonuçlar elde edilmiştir.

Anahtar Kelimeler: Meta-karmaşıklıklar, uzman karışımları, kolektif ve meta öğrenme

CHAPTER 1

INTRODUCTION

1.1 Literature Review

Our research is based on Meta Learning and Ensemble methods which have been proposed for more accurate classification, prediction and regression in Machine Learning, Pattern Recognition and Data Mining disciplines. In this study, we have analyzed the characteristical aspects and meta-features of the discretized sub regions in a dataset in order to acquire better learning performance. This literature survey in this research is categorized into three groups. The first one is about the analysis of local meta-features affecting the performance of a learner. The second one is about the learning sub regions with mixture of experts. Finally the last one is about a locally adaptive base learner classifier.

1.1.1 Survey on performance prediction of a learner

In the field of machine learning, it is impossible to know wholly which classifier outperforms or fails on a dataset. In the last decades, a new approach called "Meta Learning" has been recently proposed that performs automatic recommendation of classification algorithms based on data set characteristics and classifier selection mechanisms [1], [2], [3].

Decision Tree (DT) method, as a common classifier, is selected in order to estimate its performance by using Meta Learning. Meta Learning is accepted as *learning to learn* activity. In this method, examining the characteristical aspects of a dataset makes the classifier selection procedure easy [4]. Meta Learning, as a Machine Learning technique, automatically improves the learning method by using the experiences [5].

Meta Learning aims automatic learning that becomes flexible in solving different kinds of learning problems in order to improve the performance and the accuracy rate [6].

Meta Learning can be applicable to any field such as classification, regression and optimization. In this area there are many studies [7], [8], [9], and [10] regarding the techniques of selecting algorithm and parameter for Meta Learning. Previous Meta Learning studies [11] substantially involve meta-classification problems. Amasyalı and Ersoy [12] have made a detailed research that divides the Meta Learning activity into categories. According to the technical report study, different types of used meta-features are categorized into these classes: statistical, informational and theoretical features, subsampling landmarks features, and DT features. About 300 meta-features are aggregated into their study and used in Meta Regression model.

1.1.2 Survey on learning sub regions with Mixture of Experts

Mixture of Experts (MoE) is an extension of radial basis functions and introduced by Jacobs at el [13]. MoE locally decomposes a dataset into less complex sub-regions for better prediction performance.

In the last decades, several approaches have been proposed [14], [15]. Jacobs has proposed a model, named as “Hierarchical Mixture of Experts” [16]. Yuksel et el [17] and Masoudnia et el [18] and recently has made a thorough research, regarding the studies in Mixture of Experts. In their papers, comprehensive survey of the Mixture of Experts is provided by discussing the fundamental models for regression and classification processes. The improvements on the Mixture of Experts model using the mixtures of Gaussian process experts are also presented. Alternative localized MoE training and variational learning methods are detailed. Additionally, finding the optimum number of experts, different classification models, statistical properties of Mixture of Experts and several empirical applications in regression and classification are listed in their surveys.

1.1.3 Survey on locally adaptive base learners

There are two main empirical evaluation approaches about setting the appropriate k value for a k-NN learner on a particular dataset. The K fold cross validation process is the first simplest choice. Using some kind of validation process for different *k* values, the best one that gives the highest accuracy might be selected. The widely used second approach is the bootstrapping technique. It uses sampling with replacement to form the training set [9]. The optimal *k* value is determined via bootstrap method. Both approaches give approximately the same results. Although there are some suggestions [19] about setting the *k* value to the square root of the number of all training patterns, it is theoretically an upper bound value that limits these kinds of evaluations.

Ozger and Amasyalı [20] proposed an approach assigning the appropriate k value for a particular dataset by means of Meta Learning method. In their study, 16 meta-features are extracted from each of 200 datasets. The *k*-NN algorithms with different *k* values are computed with these datasets. In the construction of Meta training dataset, the k value giving the highest accuracy becomes the output and the extracted meta-features are accepted as attributes. It nearly becomes possible to predict the *k* value for a specific dataset by means of this new Meta training dataset. However, the biggest barrier in front of the study is the assignment of the same *k* value to the whole datasets. Moreover, the highest accuracy for more than half of the datasets is computed where the *k* parameter is 1. For that reason, regression becomes difficult.

In another research [21], some non-parametric *k* Nearest Neighbors where the general *k* for a dataset is automatically determined by geometric relationships is proposed. Classification is done by means of the centroids which globally represent the classes. Increasing the *k* Nearest Neighbors performance is obtained by the estimation of the optimal k parameter or making the *k* Nearest Neighbors algorithm adaptive to data by means of determining local decision boundaries.

Ghosh [22] and Guo et al. [23] have proposed some techniques finding a globally adaptive k value for a dataset. On the other hand, in another research [22], Ghosh has presented a locally adaptive nearest neighbor classification technique, where the value of k is automatically selected depending on the distribution of competing classes in the vicinity of the test point to be classified. The distribution of the nearest samples has a great importance in this technique.

1.2 Aims of the Thesis

A collection of observations in a dataset might have some inner complex parts. Each part in the same dataset might be statistically different from the others in various aspects. Hence, this type of a dataset needs to be decomposed into less complex sub regions. For this purpose, each sub region which has some individual and different specifications from others should be accepted as a unique dataset. In this thesis, we have aimed to analyze elaborately the characteristics and meta-features of a dataset by proposing some novel mechanisms that handle the regions of the dataset separately so that the overall learning performance will be boosted.

1.3 Hypothesis

For the purpose of accurate classification performance, we assume that it is apparently better to analyze the hidden aspects of the sub regions in a dataset, and to build a fitted mechanism that handles these regions separately by assigning localized experts. In order to prove the hypothesis, three different studies have been done. Firstly, a detailed performance analysis of a base learner (e.g., a decision tree) over any type of dataset was performed in order to understand the reasons of failure and success. Secondly, these discretized sub regions might be learned by using the Mixture of Experts model in order to enhance the overall prediction accuracy. Thirdly, a localized base learner using a dynamic parameter creator mechanism can be established to gain better performance. These hypotheses might be clarified with the supports of the detailed information given below.

Rough external specifications of a dataset such as number of samples, number of attributes and number of class labels are accepted as insufficient meta-attributes in the Meta Learning procedure. These types of specifications indicate only the density of the dataset and give little information about the data set. There are some other features such as statistical and informational theoretical features, subsampling landmarks features [24], and DT features defined in the study. Apart from these measures, there is a detailed and effective study where there is an emphasis on geometrical characteristics as a class distribution [58]. Labeled classes in a dataset can be separated or interleaved, data can be linearly separable, or regions can overlap. In this study, these kinds of measures and some others are assumed to affect the performance of the learner.

Additionally, some types of clustering methods like EM (Expectation and Maximization) [73] and DBSCAN [25] cannot handle all types of datasets. These soft (fuzzy) and distribution based types of clustering algorithms is a big barrier in front of some learning activities such as Mixture of Experts. In the traditional form of this technique, a dataset is divided into sub regions by a soft clustering method. In contrast to the soft clustering, a hard clustering method is proposed here and different gate functions are built in the process of combining the decisions of experts. Each object certainly belongs to only one cluster in hard clustering. This proposed method makes Mixture of Experts computationally possible for any type of dataset.

As a popular lazy learning technique, the *k* Nearest Neighbors classifies the test sample with a particular class by the majority voting of the *k* closest training samples. This memory based classification algorithm is used with a constant *k* value defined by the user's preference. It is generally difficult to determine the best *k* value. In the literature it is commonly recommended to assign the best *k* value for a dataset by carrying out some experiments. A constant *k* value for each test instance may results low accuracy rates. A dynamic *k* value for each test instance might augment the prediction performance.

CHAPTER 2

BACKGROUND

This introductory chapter provides the background and context for the following chapters and defines some crucial information. The chapter begins with a brief review of machine learning, which is the general context for the study described in this thesis. Common architecture of ensemble methods and Meta Learning procedures are concisely presented. Then, performance measuring criterions and paired T-Test method are described in a brief manner.

2.1 **What is learning**

As a scientific discipline, Machine learning has strong ties to Artificial Intelligence (AI). It deals with the theoretic, algorithmic and applicative sides of learning from data samples. Learning from data samples means that to build a machine (computer) program that can learn to perform a task by observing samples. Typically, this software program which uses the training samples to construct a model can make reliable predictions and decisions [26].

Machine learning discipline is typically classified into two broad categories: supervised learning and unsupervised learning. The Supervised type can be a classification or a regression problem. In classification, there is a set of predefined class labels of the samples in a dataset. Labeling the query points are performed with the help of the dataset. In a regression problem, the outputs are real numbers. The unsupervised type is called clustering. Clustering is used to find hidden structure in the unlabeled dataset. Since the samples given to the learner are unlabeled, there is no learning error or reward signal to evaluate the current solution.

Decision tree learning, association rule learning, artificial neural network (ANN) [27], inductive logic programming, support vector machines (SVM), clustering, reinforcement learning are common types in Machine Learning discipline.

2.2 **Ensemble Learning**

Ensemble methods can be also called as collective learning, committee-based learning, multiple classifier systems and classifier combination. As it is seen in Figure 2.1, there is a common architecture of ensemble learning methods. There is k number of base individual learners in the network. The $\mu(x)$ function determines the final decision of the learner network. This type of learning strategy aggregates the decisions of multiple learning algorithms in order to obtain better predictive performance than any of the base learners [28] [29].

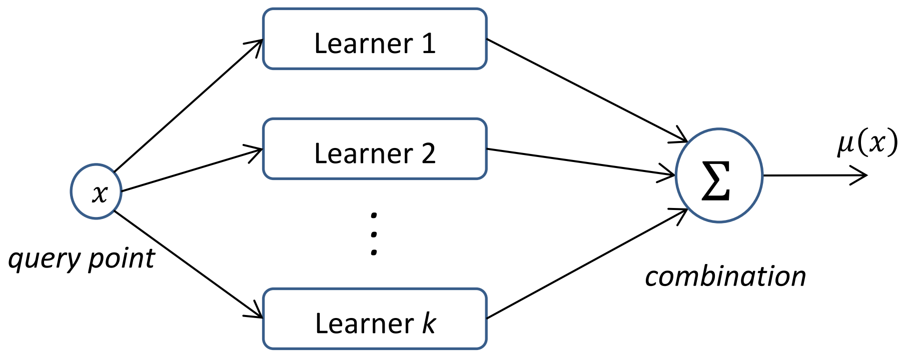

Figure 2.1 A common architecture of ensemble methods

In contrast to ordinary base learner approaches which try to learn one hypothesis from training data, ensemble methods try to build a set of hypotheses and combine them for the purpose of better prediction. It is plain that ensemble methods are able to boost the performance of base learners. In this discipline, base learners are accepted as weak learners and ensemble ones are referred as strong learners. Base learners are generally decision tree, multi-layer perceptron or other kinds of machine learning algorithms [30]. Many of the ensemble methods use homogeneous base learners inside of their mechanism. However, there are also some ensemble methods which use multiple learning algorithms to produce heterogeneous learners [31].

There are some combining methods of the decisions produced by base learners. Lior Rokach [32] and Martin Sewell [33] have presented detailed researches on these methods. Some of well-known basic methods in the research might be listed as simple uniform voting, distance weighted voting, Borda count, product, min, max, and stacking. In addition, Distribution Summation [34], Bayesian Combination [35], Dempster–Shafer [36], Naïve Bayes, Entropy Weighting [37], Density-based Weighting [38], Data Envelop Analysis Weighting Method [39], Logarithmic Opinion Pool [40], and Order Statistics [41], Stacking [42], Arbiter Trees [43], Combiner Trees [44], Grading [45] are meta-combining methods [46]. Also there are many types of ensemble methods such as Bootstrapping Aggregating (Bagging) [47], Bootstrapping Replicates (Boosting) [48], AdaBoost [49], Rotation Forest [50], Random forests [51], Stacked Generalization [52], Random Subspace Method (RSM) [53], Random Linear Oracle (RLO) [54], and Mixture of Experts [13].

2.3 **Meta Learning**

Meta Learning is a subfield of Machine Learning disciplinary. There is an automatic learning system in its mechanism derived from some empirical evaluations. The main goal in this learning style is to use some meta-features of the datasets to provide an automatic and flexible learning system in solving different kinds of learning problems. Thus, the performance of existing learning algorithms might be improved.

Meta-data actually might be external, statical, and some patterns of the datasets. Additionally, geometrical complexity measures and performance results might be added to the meta-data. Additionally, properties of the learning problem, performance measures, and patterns derived from datasets might be meta-data. These extracted features are placed into a new meta-data set for further learning activities. By using these types of meta-data, it is possible to select, alter or combine different learning algorithms in order to effectively solve a given learning problem. Hence, this discipline is regarded as "*a learning to learn*" model [55].

2.4 **Performance Measuring Criterions**

In this study, some types of measurements are used to evaluate the performances of the classifiers. Both 5x2 and 10 Ford Cross Validation methods [56] are used for accuracy measurements. The definitions are as follows:

In K Fold cross validation, the original dataset is randomly divided into equal K parts (subsamples). In the first iteration (fold), the first part is retained as the validation part; the rest of the (K-1) parts are retained as training part. In the upcoming iterations, the cross-validation process is then repeated (K-1) times. Each subsample is used for both training and validation. At the end of the folds, the K results might then be averaged in order to give a single real value between [0, 1]. The K value is substantially set to 10 [57]. When K is set to the number of samples in the dataset, it exactly becomes the Leave-One-Out cross-validation (LOOCV).

Alternately, in 5x2 Fold cross validation, the dataset is randomly partitioned into two equal parts. In the first fold, the first and the second part is retained as evaluation and train parts respectively. In the same fold, the parts then are reversed. Totally 5 folds are repeated and the results statistically are averaged.

CHAPTER 3

PERFORMANCE ANALYSIS OF A LOCAL BASE LEARNER

In the machine learning discipline, it is not possible to know entirely which classifier outperforms or fails on a dataset. There is no single learning algorithm that performs better than others for all datasets. Normally, it is recommended to test a group of selected classifiers on a dataset and choose the best one. The "Meta Learning" approach has been recently suggested in the last decades to replace the test process choosing the best one in the algorithms list. In this field, there are some studies about automatic recommendation of classification algorithms based on data set characteristics and classifier selection mechanisms [1], [2], and [3]. These types of studies mainly focus on selecting the best classifier that gives the highest classification accuracy level. However, our study aims to predict the performance of a DT classifier and to present the reasons why and how DTs fail or outperform on a dataset.

Decision and regression trees have an overwhelming fame in classification. As the Decision Trees (DTs) are fast in prediction and easy to use, it is usually preferred to the other classifiers without comparison. ID3, C4.5, C5.0 and CART are common DT types. Like the other classifier methods DTs are sensitive to datasets. A DT is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node at the bottom represents a predefined class as a final decision. On the other hand, regression trees predict the numerical value of the new instances according to its model.

DTs as a common classifier method have been selected in the study. In order to estimate the performance of a DT before using it, we have used Meta Learning. Meta Learning is accepted as a *"learning to learn"* model. In this model, examining the characteristics of a dataset makes the classifier selection procedure easy [4] and automatically improves the learning phase [5]. Meta Learning is aimed automatic learning that becomes flexible in solving different kinds of learning problems in order to improve the performance and the accuracy rate [6].

Meta Learning can be applicable to any field such as classification, regression and optimization. In this area there are many studies such as [7] [8], [9] and [10] about the techniques of selecting algorithm and parameter for Meta Learning. Previous Meta Learning studies generally involve meta-classification problems. Amasyali and Ersoy [12] have made a detailed research that divides the Meta Learning activity into categories. According to the technical report study, different types of used meta-features are categorized into these classes: statistical, informational and theoretical features, subsampling landmarks features, and DT features. About 300 meta-features are aggregated into their study and used in Meta Regression model.

This chapter has five main sections. First, it defines the problem and scope of the study. Then, the detailed definitions of the meta-complexity measures have been given. Next, there are some experimental results from the built dataset in the third section. In the fourth section, there are some techniques modeling the Meta Learning activity over the dataset. In the same section, the effects of each meta-feature on the performance level of the DTs are placed. In the last, section conclusion and future work are presented.

3.1 MEASURES OF DATA COMPLEXITY

Rough external specifications of a dataset such as number of samples, number of attributes and number of class labels are accepted as insufficient meta-features in the Meta Learning procedure. These types of specifications indicate only the density of the dataset and give little information about the data set. There are some other features such as statistical and informational theoretical features, subsampling landmarks features, and DT features defined in the study [12]. Apart from these measures, there is a detailed and effective study [58] emphasizing the geometrical characteristics of the class distribution.

The detailed version of fourteen geometrical measures has been described in the journal [58] and in the documentation [59]. Descriptions of the measures which have been as meta-features used in our study are as follows:

3.1.1 **Measures of Overlaps in the Feature Values from Different Classes (F1, F1v, F2, F3, F4)**

In this section, five types of discriminative power of the attributes in a dataset will be described below. These measures focus on the capacity of the attributes to separate samples of different classes in a future space. For each individual feature, the range and spread of the values of instances of different classes are examined. In addition, it is checked whether it is the discriminant power of a single attribute or a combination of them. In the collection, there are the following measures: the maximum Fisher's discriminant ratio (F1), the overlap of the per-class bounding boxes (F2), and the maximum (individual) feature efficiency (F3). Additionally, there are some two extra measures based on the previous ones: the directional-vector maximum Fisher's discriminant ratio (F1v), inspired by F1, and the collective feature efficiency (F4), inspired by F3. These extra meta-features are designed by [58] and [59].

3.1.1.1 **Fisher's discriminant ratio (F1)**

As a worldwide measure, the Fisher's discriminant ratio examines the maximum discriminative power of each feature. F1 is calculated as:

$$F1 = max_{i=1}^{l} f_i \qquad (3.1)$$

where l is the number of attributes. f_a is the discriminant ratio of each attribute and calculated as:

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 - \sigma_2^2} \qquad (3.2)$$

where $\mu_1, \mu_{2,} \sigma_1^2, \sigma_2^2$ are respectively the means and variances of the two classes for an attribute. Actually, μ_1 and μ_2 are the medoids representing the attributes in the dataset. f is one feature dimension. For a multidimensional dataset, the maximum f over all the feature dimensions will be chosen and accepted as F1. A high value of F1 shows that at least one of the attributes allows the separation of samples of different classes with partitions which are parallel to an axis of the feature space.

3.1.1.2 **Directional-vector maximum fisher's discriminant ratio (F1v)**

F1v derives from F1. This measure is a complement of F1 with an oriented vector that is able to separate samples of different classes. It computes the two-class Fisher's criterion that takes the following form [60].

$$R(d) = \frac{\left[\vec{d}^2(\vec{\mu}_1 - \vec{\mu}_2)\right]^2}{\vec{d}^T \overline{\sum \vec{d}}} = \frac{\vec{d}^T B \vec{d}}{\vec{d}^T \overline{\sum \vec{d}}} \qquad (3.3)$$

where $\vec{d}$ and $\vec{\mu}_i$ are the directional vector and the mean vector respectively. $\sum_i$ is the scatter matrix of patterns for class c_i. And B is found by:

$$B = (\vec{\mu}_1 - \vec{\mu}_2) * (\vec{\mu}_1 - \vec{\mu}_2)^T \qquad (3.4)$$

B is between class scatter matrix. The $\vec{d}$ is computed as

$$\vec{d} = \overline{\textstyle\sum^{-1}}\Delta \qquad (3.5)$$

where Δ is the difference between:

$$\Delta = \mu_1 - \mu_2 \qquad (3.6)$$

Also $\overline{\sum^{-1}}$ is computed as the pseudo inverse of $\overline{\sum}$ [61], [62]. If this measure is high, that means there is a vector which can separate samples from different classes.

3.1.1.3 **Volume of overlap region (F2)**

This measure computes the overlap of the tails of distributions defined by the instances of each class. For each attributes in the dataset, F2 finds the ratio of the width of the overlap interval to the width of the entire interval. The interval, mentioned here, has instances of both classes. The measure returns the product of the ratios which have been calculated for each attribute. In other words, it is the overlap of the tails of the two class-conditional distributions. For each attribute, the maximum and the minimum values of each class, and then calculating the length of the overlap region normalized by the range of values spanned by both classes.

We multiply the ratio thus obtained from each feature dimension to obtain a measure of the volume of the overlap region (normalized by the size of the feature space). Formally, let the maximum and minimum values of each feature f_i in class c_j be $\max(f_i, c_j)$ and $\min(f_i, c_j)$, then the overlap measure F2 is calculated as

$$F2 = \prod_{a=1}^{l} \frac{\mathrm{MIN\_MAX}_a - \mathrm{MAX\_MIN}_a}{\mathrm{MAX\_MAX}_a - \mathrm{MIN\_MIN}_a} \tag{3.7}$$

$$\mathrm{MIN\_MAX}_a = \min(\max(f_i, c_1), \max(f_i, c_2)) \tag{3.8}$$

$$\mathrm{MAX\_MIN}_a = \max(\min(f_i, c_1), \min(f_i, c_2)) \tag{3.9}$$

$$\mathrm{MAX\_MAX}_a = \max(\max(f_i, c_1), \max(f_i, c_2)) \tag{3.10}$$

$$\mathrm{MIN\_MIN}_a = \min(\min(f_i, c_1), \min(f_i, c_2)) \tag{3.11}$$

where l is the number of attributes, f_i is the i^{th} feature, c_1 and c_2 are the class names, $\max(f_i, c_j)$ and $\min(f_i, c_j)$ are the maximum and the minimum values of the feature f_i for class c_j respectively.

F2 will be computed by the sum of all F2 values of each pair of classes. But in our study, there is no need for these calculations because the collection of used datasets has binary classification problems.

The volume will be zero as long as there is at least one dimension and the classes do not overlap in this dimension.

3.1.1.4 **Maximum feature efficiency (F3)**

F3 describes how much each attribute contributes to the separation of the samples belonging to the both classes. F3 gives the value of the attribute which can discriminate the number of training samples.

In a region of the dataset there may be some samples from both classes. This is called over-lapping region. For each attribute, it is considered that the over-lapping region and return the ratio of the number of instances that are not in this over-lapping region to the total number of samples. This ratio is accepted as the F3 measure. [63]. In this measure it is considered only separating hyperplanes perpendicular to the feature axes. Because of this, if there is a linearly separable problem, F3 may be less than 1 if the hyperplane is oblique.

3.1.1.5 **Collective feature efficiency (F4)**

F4 is similar to F3. But F4 deals with the discriminative power of all the attributes. Because of this it is called *collective* feature efficiency. These steps below are followed to compute F4 [59]:

1. Select the most discriminative attribute that can separate a major number of samples of one class.
2. All the samples that can be discriminated are removed from the dataset.
3. The following most discriminative attribute (regarding the remaining samples) is selected.
4. If all the samples are not discriminated or all the attributes are not analyzed go to the first step.
5. The proportion of samples that have been discriminated will be returned.

Hence, the fraction of examples whose class could be correctly predicted gives us an idea. As mentioned above this is done by building separating hyperplanes that are parallel to one of the axis in dataset.

In some aspects F4 measure is different from the F3 measure, maximum feature efficiency. It deals with only the number of samples discriminated by the most discriminative attribute. Nonetheless, F4 deals with the all attributes for *collective discriminative power* measure. Because of this, F4 provides more information.

3.1.2 **Measures of Class Separability (L1, L2, N1, N2, N3)**

In this section the shape of the class boundary will be inspected by five measures in order to estimate the complexity of separating samples of different classes. In other words, these measures in this section estimate to what extent the classes are separable. The estimation is done by inspecting the length and the linearity of the class boundary. The library presents the following complexity measures: the minimized sum of the error distance of a linear classifier (L1), the training error of a linear classifier

(L2), the fraction of points on the class boundary (N1), the ratio of average intra/inter class nearest neighbor distance (N2), and finally the leave-one-out error rate of the one-nearest neighbor classifier (N3).

3.1.2.1 **The minimized sum of the error distance of a linear classifier (L1)**

This measure can be applied only to two-class datasets. L1 evaluates to what extend the dataset is linearly separable. Zero value of L1 shows that the data belonging to a particular class is linearly separable. L1 value includes the sum of the differences between the prediction of a linear classifier and the actual class value.

As linear classifier [59] a support vector machine (SVM) [64] with a linear kernel is used. This SVM is trained with the sequential minimal optimization (SMO) algorithm [65] to build the linear classifier. As it is known the SMO algorithm gives an efficient training method and the result is a linear classifier that separates the instances of two classes by means of a hyperplane. Because of this, this learner classifier has been selected.

3.1.2.2 **The training error of a linear classifier (L2)**

This measure also can be applied only to two-class datasets. In order to calculate L2 value, firstly the linear classifier (L1) is established. The training error rate is calculated by this classifier.

3.1.2.3 **The fraction of points on the class boundary (N1)**

This measure of N1 gives an estimate of the length of the class boundary in the dataset. In order to evaluate this measure firstly a Minimum Spanning Tree (MST) should be established among all the observations. Each point is connected to the nearest one. Kuruskal or Prim algorithms can be applied for the purpose of MST calculation. After building the MST that connects nodes to each other with minimum cost, the N1 ratio is found as the number nodes of the spanning tree that are connected and belong to different classes over the total nodes in the data set.

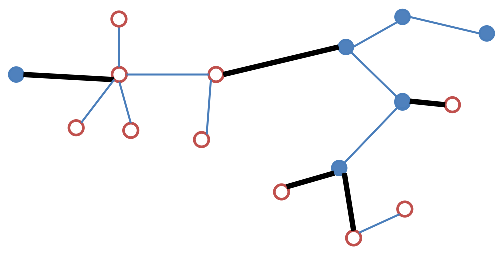

Figure 3.1 A MST connecting all nodes, darker edges connect two different classes.

N1 ratio concisely is calculated as number nodes that are adjacencies in the darker edges to the total nodes. In the figure above, N1 is equal to$10/16$.

Low values of N1 shows that it may be easier for the classifier to define this class boundary. High values of N1 show that the majority of the points lay closely to the class boundary. Hence, it will be hard for the classifier to define the class boundary accurately.

3.1.2.4 **The ratio of average intra/inter class nearest neighbor distance (N2)**

This measure compares the within-class spread with the distances to the nearest neighbors of other classes. For each input instance x_i, we calculate the distance to its nearest neighbor within the class$(intraDist(ex_i))$ and the distance to its nearest neighbor of any other class $(intraDist(ex_i))$. Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{N_e} intraDist(ex_i)}{\sum_{i=0}^{N_e} interDist(ex_i)} \tag{3.12}$$

where N_e is the number of examples in the data set.

Low values of this measure suggest that the examples of the same class lay closely in the feature space. High values indicate that the examples of the same class are dispersed.

3.1.2.5 **The Leave-One-Out error rate of the 1-Nearest-Neighbor classifier (N3)**

As it is known, in Leave-One-Out Cross Validation (LOOCV) an observation is taken out from the training data and used to have itself to be predicted by the remaining data. The prediction process is done with the 1NN classifier. This process is repeated for all the observations in the dataset for the purpose of calculating the mean square error (MSE) or the accuracy rate.

Briefly N3 is the mean square error of 1NN classifier using LOOCV. N3 measure shows how close the examples of different classes are. Low value of N3 shows a wide gap in the boundaries of classes. Conversely, high value of N3 shows low gap.

3.1.3 **Measures of Geometry, Topology, and Density of Manifolds (L3, N4, T1, T2)**

In this section some measures described in the journals [66], [67], [68], [69] are the descriptors for the geometry of the manifolds spanned by each class. These measures indirectly characterize the class separability by assuming that a class is made from single and multiple manifolds that form the distribution of the class [59]. In other words, an indirect characterization of the class separability might be provided by these measures. The problem probably is composed of several manifolds spanned by each class in a dataset. The position, shape, and interconnectedness of these manifolds give some information on how well the classes are separated and on the density or population of each manifold. The collection offers the following measures: firstly the nonlinearity of a linear classifier (L3), secondly the nonlinearity of the one-nearest neighbor classifier (N4), then the fraction of maximum covering spheres (T1), and lastly the average number of points per dimension (T2) [70], [58].

3.1.3.1 **The nonlinearity of a linear classifier (L3)**

This measure that can be implemented only for two-class data sets has proposed by [71] findings nonlinearity of a linear classifier. Given the training data set, the algorithm creates a new test set by linear interpolation with random coefficients between pairs of randomly selected instances of the same class. Then, the algorithm gives the test error rate of the support vector machine classifier trained with the original dataset. The measure is very sensitive to the overlap on the convex hull of the classes and the smoothness of the classifier boundary [59].

3.1.3.2 **The nonlinearity of the 1-Nearest-Neighbor classifier (N4)**

N4 creates a test set as proposed by the nonlinearity of a linear classifier (L3) and returns the error rate of the one-nearest-neighbors (1NN) classifier.

3.1.3.3 **The fraction of maximum covering spheres (T1)**

This measure has been proposed in the study [72]. T1 describes the shapes of class manifolds with the opinion of *adherence subset* as it is seen in figure below. In other words, an *adherence subset* is a sphere centered on a sample of the data set which is grown as much as possible before touching any samples from the other class. Thus, an adherence subset includes a set of samples from the same class and it cannot grow more without including samples out of the class. The measure considers only the biggest adherence subsets or spheres by removing all those that are included in others. Then, the measure gives the number of spheres normalized by the total number of points [59].

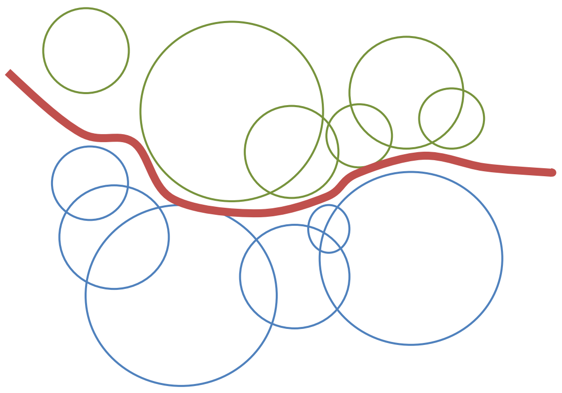

Figure 3.2 Retained adherence subsets for two classes near the boundary.

3.1.3.4 **The average number of points per dimensions (T2)**

As a rough indicator metric, this measure gives the density of the dataset. T2 indicates the ratio of the number of samples in the dataset to the number of the attributes. In other words, if the dataset is regarded as a matrix, F2 is calculated as the ratio lines to columns.

The list of the 14 geometrical characteristics describing meta-features are in the Table 3.1 below.

Table 3. 1 List of Used Measures as Meta-features

No.	**Abbr.**	**Description**
1	F1	Maximum Fisher’s discrimination ratio
2	F1v	Directional-vector Max. Fisher’s discriminant ratio
3	F2	Volume of overlap region
4	F3	Maximum (individual) feature efficiency
5	F4	The collective feature efficiency
6	L1	Minimized error by Linear Programming (LP)
7	L2	Error rate of linear classifier by LP
8	L3	Nonlinearity of classifier by LP
9	N1	Fraction of points on boundary (Minimum Spanning Tree method)
10	N2	Ratio of average intra/inter class NN distance
11	N3	Leave One Out Error rate of 1NN classifier
12	N4	Nonlinearity of 1NN classifier
13	T1	Fraction of points with associated adherence subsets retained
14	T2	Average number of points per dimensions

3.2 **Computational Cost of the Complexity Measures**

Computational time complexities of the complexity measures described above can be seen in Table 3.2. n is the number of input samples, n_t is the number of test samples (applicable only to the measures that generate an additional test set), c and a are the number of classes and attributes respectively. The time complexity of $O(SMO)$ is to build a SVM with linear kernel by means of the sequential minimal optimization (SMO) algorithm.

Table 3. 2 Computation costs of complexity measures

No.	Measures Labels	Time Complexities
1	F1	$O(n.a)$
2	F1v	$O(n.a + a^3)$
3	F2	$O(n.a)$
4	F3	$O(n.a.c)$
5	F4	$O(n.a^2.c)$
6	L1	$O(SMO)$
7	L2	$O(SMO)$
8	L3	$O(SMO + n_t.a.c)$
9	N1	$O(n^2.a)$
10	N2	$O(n^2.a)$
11	N3	$O(n^2.a)$
12	N4	$O(n_t.a.c + n.n_t.a)$
13	T1	$O(n^2.a)$
14	T2	$O(1)$

3.3 EXPERIMENTAL STUDY

The procedure of our empirical study has the following steps: preparing datasets, extracting the meta-features from the datasets, computing the accuracy rates of the DTs, building a linear regression model according to the meta learning dataset, and finally predicting the accuracy rate of a new dataset according to the linear regression model. After computing the linear regression model, it becomes possible to understand how and why the meta-features defined in this article affect the accuracy of a DT. In other words, it will be available to explain in which circumstances DTs outperform or fail on a dataset.

3.3.1 Preparation of Test Datasets

The scope of our study covers 115 real-world two-class datasets taken from the UCI benchmark data repository [73]. We have selected only two-class problems as most of the meta-features defined above can be only applicable to two-class datasets. 42 of them are originally two-class datasets. 73 of them have been transformed into two-class datasets artificially from the UCI multi-class datasets. The transformation is done for the purpose of enriching the meta dataset.

The conversion of a multi-class data set into a two-class dataset has been done by discriminating one of the classes against the others. For example, it can be produced *m* two-class datasets from a dataset in which there is *m*-class labels. We have selected few of the re-created datasets, because most of them are skewed datasets. The re-created datasets are selected if two of the class distributions are similar in amount. There is an important point that should be taken into consideration, in a skewed two-class dataset if one of the class distributions is about 95% of all, the accuracy rate will be 95% according to the Zero Rule, a random prediction method. As the most important parameter of performance, the high accuracy rate which is close to 1.0 misleads us to some useless inferences. In case of unequal distribution of classes in skewed datasets, the calculated meta-features will damage the learning activity.

In the original type of UCI datasets, some instances have missing values; some of the values are nominal. Before using them we have done some changes in the Weka software platform without damaging their originalities. Missing values have been replaced; nominal values have been converted to numerical values and finally all the values are normalized. The file format of the datasets is KEEL [74] that is an extended form of the ARFF [75] format. Each dataset in our study has two forms as KEEL and ARFF. KEEL is used in both DCoL [59] and the MATLAB software, and ARFF is used in Weka.

3.3.2 **Preparation of Meta Dataset**

14 meta-features from the datasets have been extracted in order to use for Meta Learning. These 14 meta-features have been computed by means of the DCoL software which proposed in the document [59]. The DCoL written in C++ has been compiled and implemented in the Linux operating system. 14 computed meta-features are the attributes of the new meta-dataset. The outputs of the new dataset are the accuracy rates taken from the DT algorithm. The accuracies have been calculated by using CART algorithm in MATLAB by using 10 fold cross validation. We have applied Linear Regression to the meta-training set (shown at the Table-4) by using 10 fold cross

validation in Weka. The actual and the predicted accuracy rates by means of 10 fold cross validation have been computed. The rates differ from 0.5644 to 1.0 as they are seen in the Figure-3.3. All the computational results are placed in the Appendix-A at the end of the thesis.

All the data in the meta-learning training datasets have been normalized as far as possible for comparability across problems. Thus, the ranges of values are taken to the interval between 0 and 1. Only the class labels have not been normalized because the accuracy rates should be remained original.

3.4 PREDICTION OF DT ACCURACY

Linear regression algorithm in Weka has been run on the meta dataset in order to find the contributions of meta-feature on the decision making activity. Different combinations of meta-features have been evaluated on the dataset and taken different results in the tables at the Appendix A. In the table, there are four kinds of outputs: the DT accuracy formula, the correlation coefficient score, the root mean squared error (RMSE), and the mean absolute error (MAE) [76]. In each line the outputs are calculated according to the given parameters. The Correlation coefficient metric can be used as a type of performance criteria here. Its value varies from -1 to +1. -1 indicates perfect negative correlation, 0 indicates no correlation, and +1 indicates perfect positive correlation. Close value to 1 can be regarded as better prediction accuracy. Besides these criterions, the number of hyper parameters considerably plays a great role. Obviously less parameter is always better according to Occam's razor [77].

As a brief definition, the RMSE rates which is computed by the difference between the actual and the predicted values are shown in the Figure 3.3 as a green line. The mean absolute error is an average of the absolute errors. It is a quantity used to measure how close predictions are to the eventual outcomes. The RMSE and MAE formulas are given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2} \tag{3.13}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - t_i| \tag{3.14}$$

where *N* is the number of samples. y_i and t_i define the actual value and the estimated values of the i^{th} sample, respectively. Both RMSE and MAE measures are used together to diagnose the variation in the errors in a set of forecasts. RMSE is always larger or equal to MAE. If there is equality between them, it means all the errors are of the same magnitude. If there is greater difference, it means there is a greater variance in the individual errors.

As the meta learning dataset is seen as a normal training dataset, some other algorithms have become applicable. 1NN, Multilayer Perceptron, REPTree and Bagging have been run over the dataset and taken the correlation coefficient results 0.8558, 0.8332, 0.8647, and 0.9054 respectively. This means that this linear regression model gives the best classification performance of all.

Table 3. 3 DT Accuracy Formula with different measures

	Given Parameters	DT Accuracy Formula	Corr. Coef.	MAE	RMSE
1	F1, F1V, F2, F3, F4, L1, L2, L3, N1, N2, N3, N4, T1, T2	$0.056 * \mathrm{F3} - 0.04 * \mathrm{F4} - 0.106 * \mathrm{L2} - 0.3 * \mathrm{N3} + 0.9986$	0.9207	0.0326	0.0472
2	L2, N3, N1	$-0.115 * \mathrm{L2} - 0.299 * \mathrm{N3} + 0.9985$	0.9245	0.0305	0.0459
3	L2	$-0.3653 * \mathrm{L2} + 0.9723$	0.7482	0.0699	0.0801
4	N3	$-0.3664 * \mathrm{N3} + 0.9744$	0.8998	0.0319	0.0506

As it is seen in Table 3.3, the linear regression formulas are computed according to the normalized dataset. In these formulas, the coefficient values for each attribute should be able to be compared about the rates of effects. By this normalization process the rates of effects in the regression formula can be discussed easily. Moreover the coefficient values are used to retrieve plausible and rational information, regarding the reasons why and how DTs outperform or fail.

Firstly, we have tested on the meta-learning dataset including the whole 14 attributes by using linear regression algorithm as it is seen in the first row in Table 3. 3. As the correlation coefficient value is 0.9207, the classification application is accepted as successful because the result is very close to 1. Since the values of MAE and RMSE measures are very small, it can be said that there is a successful accuracy prediction of the DT classifiers on any dataset. In the linear regression formula, the constant value of 0.9986 indicates that the accuracy level will decrease due to the variables having minus coefficients (F4, L2 and N3). At this point, N3 has a great reverse influence on prediction. The higher value it is, the lower accuracy we take. N3 is three times bigger than L2 on prediction. F3 and F4 have worthless effects because of their insignificant coefficients.

Secondly, we have applied Correlation Feature Selection (Cfs) algorithm to the meta-dataset by means of Weka. The implementation of Cfs has selected only 3 attributes (L2, N1, N3) as it is seen in the second row of the table. The linear regression model contains only two parameters, L2 and N3. Applying the cross validation procedure to the new 3 dimensional dataset by using linear regression method, the Correlation Coefficient value has slightly increased. On the other hand, the MAE and RMSE scores have slightly decreased. Thus, this step can be preferred because of less complexity and less computational time according to Occam's razor. By using Correlation Feature Selection (Cfs) only two parameters (L2 and N3) are accepted as qualified in prediction. These two parameters give us two main ideas.

1. Accurate performance prediction can be available.
2. These two determinant factors affect the performance of a DT classifier on a dataset.

Thirdly, in order to reduce three things, the computational time, the complexity and the number of parameters, we have tried to predict the DT accuracy with only one parameter. The results in the third row are calculated by using only L2 score. In this step, the RMSE score increases and the correlation coefficient value decreases. These values indicate unsuccessful prediction.

In the last step only the N3 criteria is used as a parameter. Better outcomes are taken than the previous step. As it is seen below, preferring to use uniquely N3 measure rather than using L2 gives better performance. Choosing merely the L2 parameter leads us to better accuracy prediction of a DT classifier.

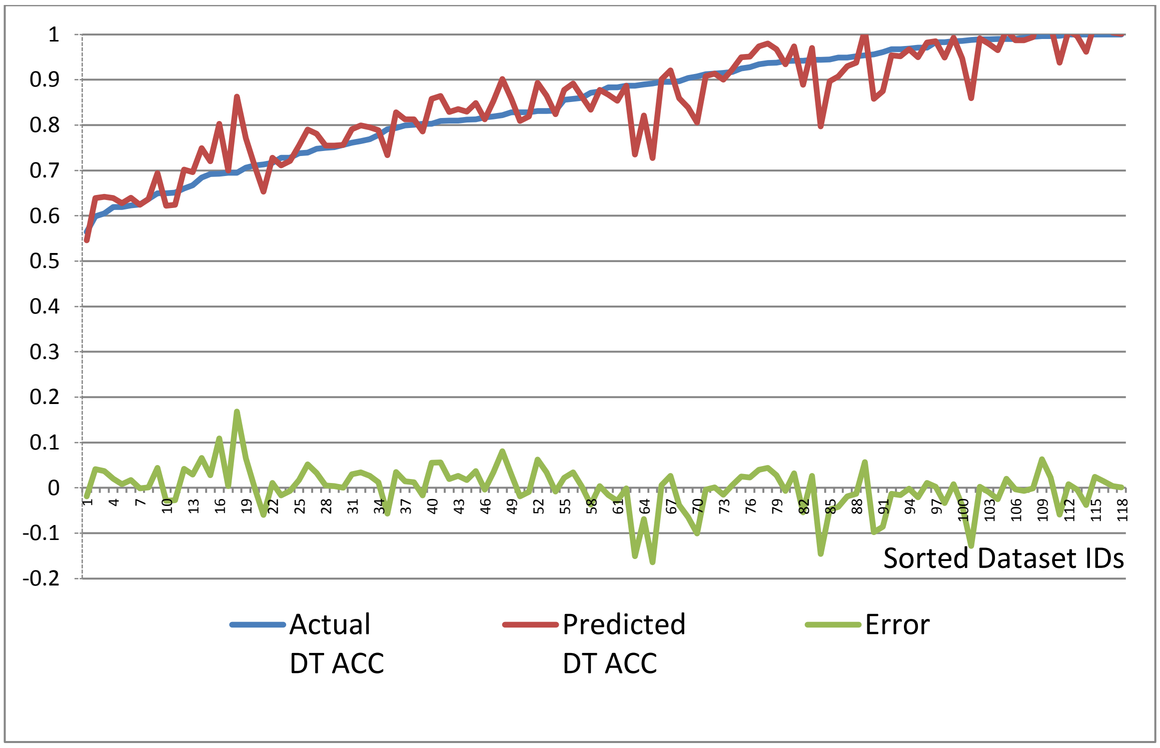

Figure 3.3 Actual and predicted DT accuracy rates with error rates

Figure 3.3 shows the accuracy rates of the DT classifier on each of the datasets in order to give a notion about the work space. Notice that the accuracy rates have been sorted for analyzing the work space easily. These three lines in the figure are actual, predicted and error rates. In the *x* line there are sorted dataset IDs according to their actual accuracy rates. Here we have to emphasize that each of the sorted dataset according to their actual accuracy rates, is not related to the next and previous neighbors in the sequence. The predicted rates are taken from the linear regression model. As it is seen the predicted values are very similar to the actual ones. The ripples in the predicted and error lines indicate the RMSE error rates. The averages of RMSE and MAE rates are 0.0326 and 0.0043 respectively. Because the error rates are tiny in amount, the performance of the system can be accepted as successful.

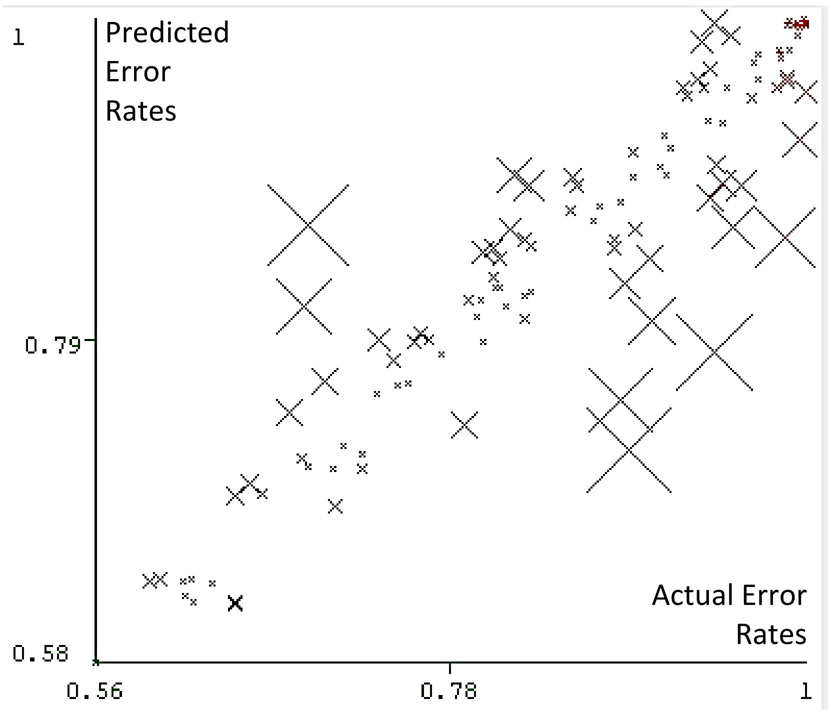

Figure 3.4 Visualized classifier errors between actual and predicted accuracies

Using the linear regression model, classification errors taken from Weka are visualized in Figure 3.4. The size of the multiplication sign, × demonstrates the magnitude of the error rate for each item in the dataset. Most of the tiny × signs lying in the diagonal line indicate very few error rates. The big × signs spreading in outer side of the diagonal line are in minority.

3.4.1 **Analysis of the Results**

With the help of the previous step, the level of performance of a DT can be easily estimated by using the linear regression model. In the second line of Table-2, the formula basically indicates that two main factors, L2 and N3 are most significant on prediction. As described in this article, briefly N3 is the Leave One Out Error rate of 1NN classifier and L2 is the Error rate of a linear classifier. N3 measure indicates how close the examples of different classes are.

As the N3's coefficient is approximately three times bigger than L2's, N3 has highest effect on prediction. From the linear regression model it is understood that when the value of N3 increases, the value of accuracy marginally decreases. In contrast, when N3 decreases accuracy increases. This criterion depicts that the higher gap in the boundaries of classes.

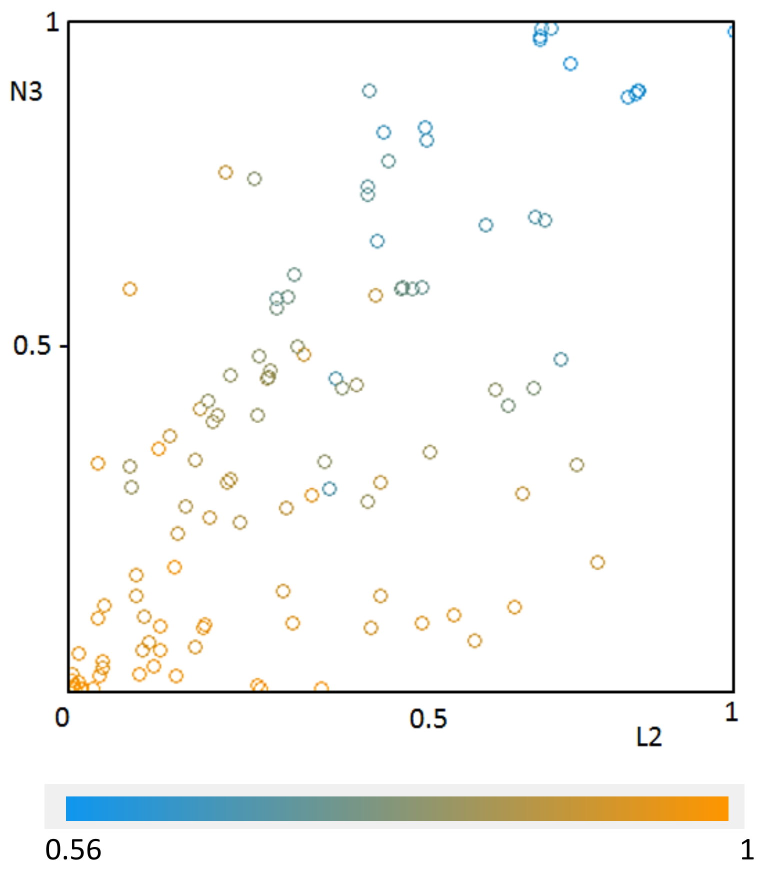

Figure 3.5 Accuracy rates according to L3 and N3

Figure 3.5 is taken from the Weka software showing the performance of the linear regression model according to the axis L2 and N3. Circle colors range from blue to orange. The darker blue circles indicate the worst prediction accuracy which is close to 0.56. The darker orange color indicates best prediction accuracy. This figure gives a simple idea, the lower value L2 and N3 are, the higher performance there is in the classification phase.

Another factor for performance criteria is L2. The training error rate, L2 is calculated by a linear classifier that provides the information about what extent the training data is linearly separable. From the regression formula it is derived that when the value of L2 increases, the value of accuracy slightly decreases and vice versa. It can be concisely said that if it is able to be built a linear classifier that bisects the class distributions with a less error rate, DTs gives better results in these circumstances.

Another method for computing the prediction with less parameter is to use only N3 criteria. Despite the fact that it does not give better prediction than the L2 and N3 combination, N3 by itself gives a notion about the performance of DTs on the particular dataset. Furthermore, since it is a single parameter, N3 can be considered as sufficient for performance prediction.

As a result, the performance of a DT is explicitly related with these two criterions, error rates of both 1NN and linear classifiers.

3.4.2 **The Correlations of Meta-features**

In this section, the correlations among the meta-features are analyzed. The highly correlated meta-feature pairs guide the Meta Learning dynamics. The correlation coefficient of each meta-feature pairs has been calculated. The meta-feature pairs are considered as correlated if the correlation coefficient's absolute values are bigger than 0.8 [12]. The number of highly correlated meta-feature pairs is shown in bold at the Table 3. 4.

The Correlation matrix in the Table-3.4 describes pair correlations among 14 meta-features including accuracy levels. In the matrix the value in the cell (i, j) is equal to the correlation coefficient between i^{th} and j^{th} features. The diagonal elements are the correlations of variables with themselves and they equal to 1. The half of the table is not given because the matrix is diagonally symmetric.

Table 3. 4 Correlation matrix of meta-features

	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2	DT Acc
F1	1	0.21	-0.11	0.40	0.24	-0.16	-0.42	-0.44	-0.44	-0.41	-0.44	-0.46	-0.23	-0.11	0.43
F1v		1	-0.10	0.07	0.14	-0.17	-0.26	-0.27	-0.22	-0.12	-0.22	-0.22	0.07	-0.04	0.23
F2			1	-0.41	-0.55	0.15	0.13	0	0.40	0.29	0.41	0.28	-0.11	0.04	-0.32
F3				1	0.78	-0.32	-0.42	-0.22	-0.65	-0.62	-0.65	-0.52	-0.32	-0.13	0.62
F4					1	-0.24	-0.31	-0.22	-0.57	-0.43	-0.60	-0.53	-0.09	-0.29	0.50
L1						1	0.40	-0.14	0.36	0.47	0.35	-0.04	0.15	0.01	-0.34
L2							1	0.59	0.70	0.49	0.69	0.53	0.15	-0.10	-0.76
L3								1	0.30	0.01	0.34	0.70	0.09	0.11	-0.38
N1									1	**0.83**	**0.98**	0.61	0.31	-0.12	**-0.91**
N2										1	**0.82**	0.34	0.47	-0.20	-0.76
N3											1	0.65	0.29	-0.09	**-0.92**
N4												1	0.12	0.23	-0.62
T1													1	-0.24	-0.28
T2														1	0.11
DT Acc															1

The highly and positively correlated meta-feature pairs are (N1,N2), (N1,N3), (N2,N3). The highly and negatively correlated meta-feature pairs are (N1, DT Acc), (N3, DT Acc).

As the measures N1, N2, and N3 are in the same scope used for the class separability, it is accepted as they affect from the same source.

N1 and N3 have the reverse effect on accuracy because of their negative correlations, (N1,Acc) and (N3,Acc) are -0.91 and -0.92 respectively. When the values of N1 and N3 decrease, the accuracy level increases and vice versa. Also in the DT accuracy formula, N3 has an overwhelming influence on prediction. In the correlation matrix table, it gives similar results. As described before, N3 is the Leave One Out Error rate of 1NN classifier, indicating how close the examples of different classes are. Low N3 value depicts wide gap in the boundaries of classes and vice versa. It means that to take high performance from a DT running over a dataset, the dataset should not include narrow class margins.

Again, N1 measure is the fraction of points on boundary (Minimum Spanning Tree method). It gives an estimate of the length of the class boundary in the dataset. Briefly, the case where the points from the same class are usually together positively and highly correlate the prediction performance.

Additionally, N2 and L2 have a slight correlation (-0.76) with DT accuracy. The effect of N2 has been previously detected in the linear regression model. But L2 appears itself here. L2, as mentioned before, gives the information about what extent the training data is linearly separable. The existence of a linear separablity with the low error rate induces better prediction with DTs.

3.5 **CONCLUTIONS AND FUTURE WORK**

The accuracy rate of a DT classifier on a certain dataset may differ from the other datasets. In this study we have focused on the performance analysis of DTs. We have tried to guess the behavior of DT classifier on a dataset. Also we have tried to find out satisfactory answers to those questions why and how DTs outperform or fail on a dataset and what kind of features contribute to the performance of a DT. For this purpose, a collection of two-class datasets is used for our empirical study. 14 meta-features have been derived from the collection. Only two features, the LOO Error rate of 1NN classifier and the Error rate of linear classifier, are highly influential on prediction activity. In other words, these two meta-features mainly affect the performance of DTs.

After building the formula of linear regression model, we have tried to understand how and why the meta-features affect the accuracy of DT. We have taken sufficient and successful results such as a meaningful and feasible regression formula including meta-features. Thus, it becomes easy to predict the performance of a DT classifier. Briefly we have concluded our study in two main ideas:

1. The higher the gap in the boundaries of classes is, the higher performance we take from DTs' predictions.
2. Building a linear classifier that bisects the class distributions with less error rate increases the performance of DT classifier in these circumstances.

In our study we have found out two main factors determining the performance of DTs. In this scope, firstly two classifiers for the factors have to be trained. These required classifiers are a kNN learner for N3 measure and a linear classifier for L2 measures. It is obvious that the cost of training both of the classifiers is not less than or equal to the DT's. In this case it would be clearly better to train only a DT classifier rather than these two. Naturally one of the purposes of the Meta Learning Discipline is to reduce the complexities and computational times in supervised learning. To predict the performance of complex classifiers such as SVM and MLP [78] by using simple classifiers is accepted as a satisfactory way. But our study differs from this point; our intension is to unveil the reasons and to find out determinant factors why DTs exceed on a dataset by examining the geometrical and hidden characteristics of the dataset. Moreover, this study gives a simple idea whether or not a DT learner is applicable and suitable for a dataset.

The accuracy rates of DTs in some re-created datasets are very close to 1 because of the unequal class distributions. The high accuracy rates become a great barrier in analyzing and evaluating the study. Few of the artificial datasets may be outlier or noise and have an effect on decreasing the performance. Removing them may increase the performance.

Some other performance metrics can be used. Even though accuracy is a common criterion for the performance of a classifier, there are some other methods such as precision, recall, F1-Score, ROC (Receiver operating characteristic) space [79], and the Matthews Correlation Coefficient (MCC) [80] metric for binary class datasets. Since F1-Score should be calculated for both of the classes, MCC plays a better role. Where the accuracy rate is close to 1 in the case of unequal distributions, MCC measure gives more realistic and reliable scores ranging from -1 to +1. In this context, MCC will be the best metric to analyze the classifier performance in further steps.

CHAPTER 4

LEARNING SUB REGIONS WITH MIXTURE OF EXPERTS

As an extension of radial basis functions, Mixture of Experts is firstly introduced by Jacobs at el [13]. In this learning system, the main purpose of MoE is to locally decompose a dataset into less complex regions by means of divide and conquer technique to obtain better prediction performance. Each sub region of the dataset is accepted as an individual unique dataset. An allocated expert (classifier) is specifically trained for its own region. The outputs of the experts are merged through a generalized linear rule called as a gating function.

Mixture of experts is a voting technique where the votes are given by the experts. Figure 4.1 presents a graphical representation of the basic MoE architecture. The architecture consists of *k* base learners, namely experts. Each localized base learner becomes an expert of its own region. Each expert has their individual decision for the test point. But the gate function only selects the related experts to classify the query point. In other words, the class prediction for the test point is performed by the gate function that combines all the decisions of the related experts. In calculations of the final decision via the gate function, the experts have different weights according to their distances to the test points. The combiner system μ also includes this gating system. The final output of the observation *x*, shortly μ is a weighted average found by this formula:

$$\mu(x) = \sum_{i=1}^{k} w_i(x)\mu_i \tag{4.1}$$

where x is the observation (query point), k is the number of the experts and w_i is the weight of the expert μ_i.

In the traditional MoE model, the gate function activates only some experts which are related with the query point. The gate function assigns different weights to the experts. The gate mechanism outputs a set of scalar coefficients denoted by w_i. The outputs of the experts, μ_i, are weighted by these gating outputs.

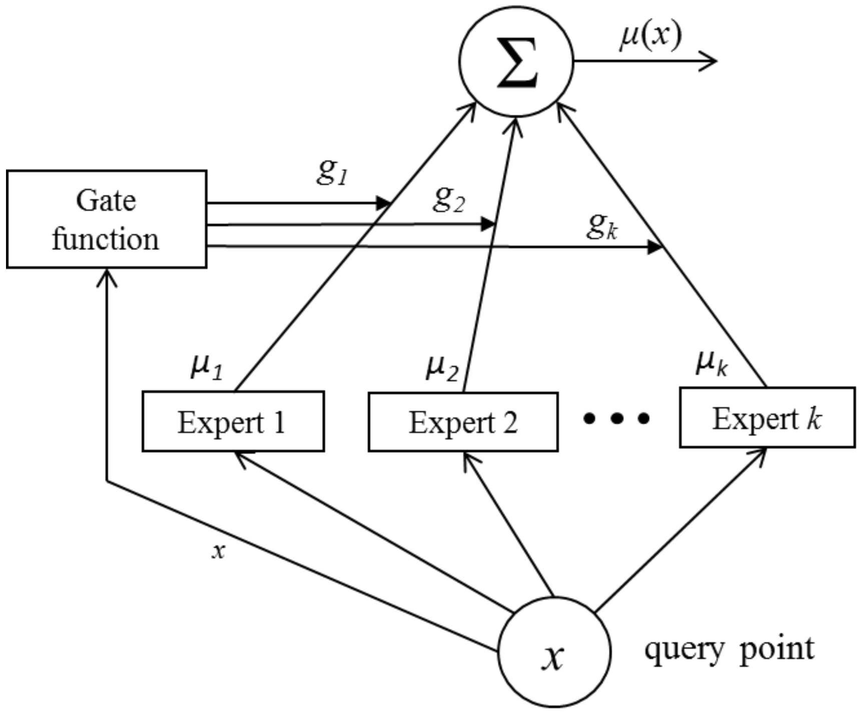

Figure 4.1 The principles of MoE Method

In the last decades, several theoretical developments, different approaches and some models have been proposed. Firstly to improve the traditional MoE model, Jacobs has proposed a model, named as “Hierarchical Mixture of Experts” [16]. Yuksel et el [17] and Masoudnia et el [18] and recently has made a detailed research about the studies in Mixture of Experts. In their papers, comprehensive survey of the MoE is provided by discussing the fundamental models for regression and classification processes. The improvements to the MoE model on the mixtures of Gaussian process experts are also presented. Alternative localized MoE training and variational learning methods are detailed. Additionally, finding the optimum number of experts, different classification models, statistical properties of MoE and several empirical applications in regression and classification are listed in their surveys.

A dataset might have complex parts and needs to be decomposed into less complex regions. Each region in the same dataset might be statistically different from the others in various aspects. These differences can be related with the number of samples, the number of classes, and the number of attributes. All of these features are the determinant factors describing the sparseness and denseness of a dataset. In addition, each part of a dataset might have individual and different specifications from others. The specifications can be data distribution type, structure, density of data, the number class labels, the separability of the class boundaries, and some other meta-features. These facilities can decrease the overall accuracy rates thanks to the generalized model for the corresponding dataset. Furthermore, there might be some other geometrical meta-features [81] affecting the performance of the classifier. For this purpose, each sub region of the dataset might be accepted as a unique dataset. In order to boost the overall performance of a classifier on the same dataset, it will be better to build a mechanism that assigns a locally adaptive expert to each region.

The performance of Mixture of Experts is strictly related with responses of the three particular questions.

1. How to divide the dataset into sections?
2. How to train the local experts?
3. How to label a query point?

As a response to the first question, Expectation Maximization (EM) algorithm is proposed in the literature [17] [82]. EM [83] is a soft (fuzzy) clustering method. Unlike many other clustering approximations, the regions have soft cluster boundaries meaning that instances might lie simultaneously in other clusters. Therefore the cluster boundaries possibly overlap since it is a statical clustering method where all instances may belong to all clusters with a given probability (e.g. a likelihood of belonging to the cluster). EM assigns each point to the most probable cluster. In statistics, the EM algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) [84] estimates of parameters in statistical models. The EM iteration alternates between two main steps: expectation (E) and maximization (M). The (E) step generates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters. The (M) step, calculates parameters maximizing the expected log-likelihood found on the (E) step. Then, these parameter-estimates are used to determine the distribution of the latent variables in the upcoming (E) step.

To the second question, it is suggested to use an effective classifier fitted to the sub region. In many empirical applications, linear regression models are generally preferred as local experts although there are some other popular local experts such as MLPs and SVMs [17].

Using a gate function integrating the decisions of the related experts with the test point is the third answer.

In this study, it is proposed to use a hard and statical clustering method rather than a soft one in order to divide the dataset into regions. It has also been suggested to build four types of gate functions combining the predictions of the experts. Although it is proposed in the literature to choose any expert for any region, only decision tree (DT) classifiers are preferred throughout the experiments due to their performance, effectiveness, and clearness and their white-box structure.

In this chapter of Mixture of Experts, there are five main sub-sections. In the first section, there are the descriptions of the proposed MoE method and the definitions of the fitted gate functions. In the second section, there are the experimental results. In the third section, there are complexity analyses of the algorithms. Finally there are some evaluations and future studies before the conclusion section.

4.1 **METHODOLOGY**

In this study, a hard clustering method is used in the procedure of dividing a complete dataset into disjoint subsets and different gate functions are built in the process of combining the decisions of experts. A dataset is divided into regions by a hard clustering method and a decision tree classifier is constructed for each region as an expert. In classical MoE approach, normally, a query point is classified by its corresponding experts.

In contrast to the soft clustering, each object certainly belongs to only one cluster in hard clustering. As one of the popular method, *k*-means clustering is a nonhierarchical method used for grouping a dataset. It groups data points using a “top-down” recursive approach with a predefined parameter, number of clusters. This method is computationally possible for any type of dataset.

Nonetheless, there are some types of popular and effective clustering methods such as EM [83], DBSCAN [85] and OPTICS [86]. Although these methods are very powerful and suitable in discretization of some types of datasets, they cannot handle all types of datasets. Density-based spatial clustering of applications with noise (DBSCAN) and OPTICS are density-based clustering algorithms. DBSCAN requires two parameters. The first one, ε (epsilon) is the maximum search radius. The second one is the minimum number of points required to form a dense region. Because of this, DBSCAN cannot guarantee to find a predefined number of clusters in the given dataset. In our empirical studies, this situation has made impossible to test the proposed MoE model.

The cost function of *k*-means is as follows: Given a set of *N* points (x_1, x_2, … x_n), *k*-means clustering method partitions the observations into K (≤ n) clusters $C = \{c_1, c_2, \ldots, c_k\}$ in order to minimize the within-cluster sum of squares (WCSS). The objective function is to find:

$$J = \underset{C}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in C_i} \|x_i - y_i\|^2 \tag{4.2}$$

where y_i is the *i*th cluster center, in other words it is the mean value of points in the cluster C_i. C_i is the input data points within the corresponding cluster. $\|x_i - y_i\|^2$ is the chosen distance measure between x_i and y_i. *J* is an indicator of the distance of the *N* data points from their respective cluster centers.

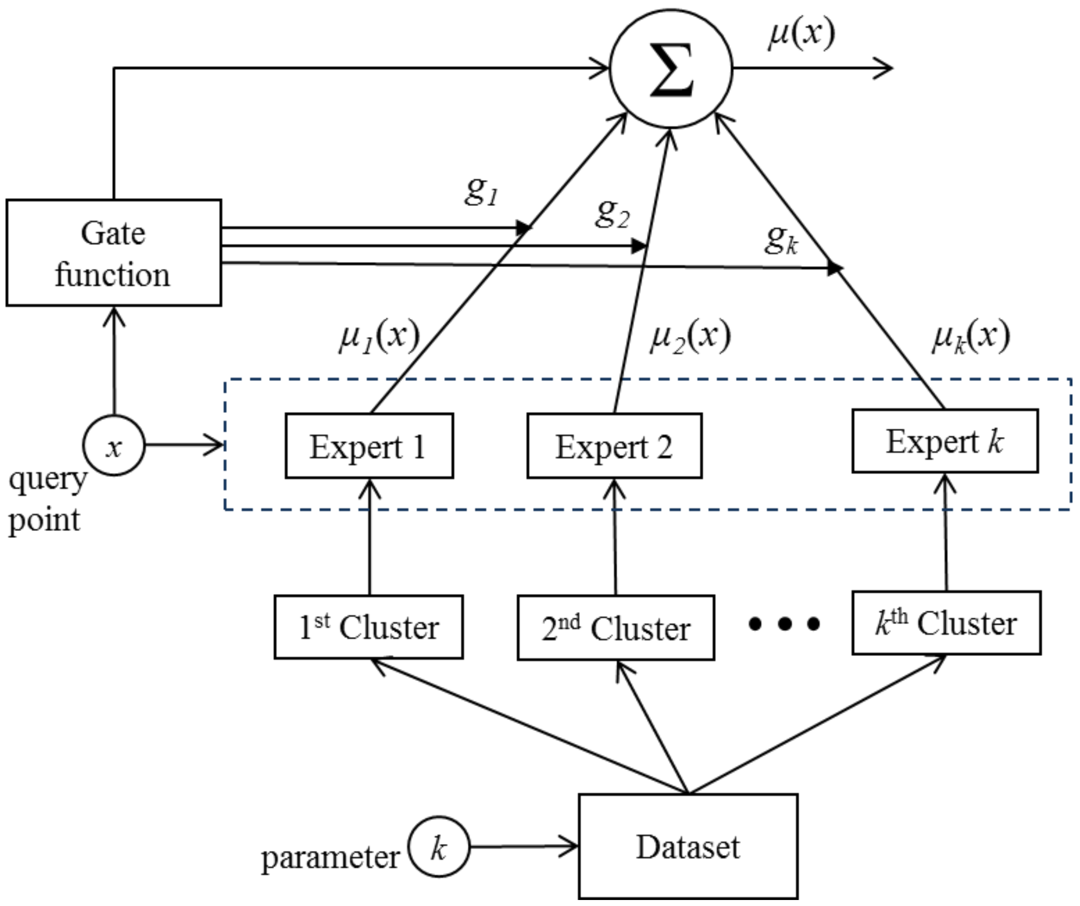

Figure 4.2 Proposed MoE method

Figure 4.2 illustrates the proposed model of Mixture of Experts. The query point *x* is classified by the gate function assigning different weights to the *k* experts in the calculation. The proposed method basically runs as follows:

1. Normalize all input patterns, divide the dataset into *k* sub regions (hard clusters)
2. Assign an expert (DT) to each region and train it
3. Ask the query point to the whole experts, compute each decision
4. If needed, Compute the distance between the test point and the central point of the *i*th expert's region as in the distance formula (4.3) below
5. Combine the decisions of the experts by one of the four different types of gate function in order to classify the query point

The distance formula (4.3) is as follows:

$$dist(x, E_j) = \left(\sum_{i=1}^{D} \left| a_i(x) - a_i(E_j) \right|^r \right)^{1/r} \tag{4.3}$$

where E_j is the central point of jth expert's region. The center of an expert's region is the average of the all member samples in the region. $a_i(x)$ denotes the value of the ith attribute of instance *x*. The arbitrary instance *x* in the dataset is described by the attribute vector: $\langle a_1(x), a(x), a(x), \ldots a_D(x) \rangle$ where *D* is the number of dimensions (attributes). In this formula, if *r* is set to 1, it becomes *Manhattan* (*Cityblock*) distance; if it is set to 2, it is Euclidean; if it is set to more than 2, it turns into *Minkowsky* distance. In the limiting case of *r* reaching infinity, *Chebyshev* distance is obtained. As an alternate and versatile choice, *Mahalanobis* is another metric between a point and a distribution. Apart from the others, *Mahalanobis* puts emphasis to the distributions. In our design, Euclidean method is used to calculate the distance between the query point *x* and the representative central point E_j.

4.1.1 **Base Learner**

In MoE, the use of DTs is firstly introduced by Jordan and Jacobs [7]. There are recently many researches [17] on MoE and proposed new approaches and showed their differences. Using MoE with Radial Basis functions is firstly presented by Lei Xu [87]. It is known that DTs are used in Radial Basis Function Networks as a DT-RBF (Decision Tree based Radial Basis Function) since DTs ease the classification phase as a compact algorithm [88]. DTs have more advantages than the other learners since they are fast in prediction. Hence, DTs are preferred to use as experts of the regions [89], [90]. In this study, decision tree classifier is preferred as a base learner. DTs employ a top-down recursive strategy for growing the trees. *k* DTs are established for *k* regions.

4.1.2 **Gating Functions**

The final prediction in the MoE ensemble method is given thorough the gate function. Therefore, the mechanism of the gate function plays an overwhelming role in the performance of the prediction. The experts can be combined using one of the proposed combination rules. In these improved ensemble mechanisms, there are four types of gate functions distributing different weights to the local experts.

The decision *d* of the i^{th} expert in the network might be defined as $d_{i,j}$ with this $\mu_i(x)$ function:

$$\mu_i(x) = d_{i,j} \in [0,1], i = 1, \ldots, K; j = 1, \ldots, C \tag{4.4}$$

where *K* is the number of experts and *C* is the set of class labels, μ_i is the decision of the *i*th expert. The decision $d_{i,j}$ which is between [0, 1] is the probability of the corresponding class label. The probabilities of all the class labels for a test point are calculated by the expert $\mu_i(x)$ and the sum should be exactly 1:

$$\mu_i(x) = \sum_{j=1}^{C} d_{i,j} = 1 \tag{4.5}$$

The probabilities of each class label are computed by all the experts in the network and placed in a K-by-C matrix for the usage of proposed algebraic combiners. These algebraic combiners are called as commensurative, Borda Count, competitive and cooperative gate functions. The combination methods accepted as generalized gating function models are listed as follows:

4.1.2.1 **Commensurative Gate Function**

In this style, all the experts have rights on the calculation and they have equal weights in the prediction. Simple majority (plurality) voting technique is applied. If there are *K* experts for a binary class problem, the ensemble decision will be the output of the $\left\lfloor \frac{K}{2} + 1 \right\rfloor$ experts which choose the same class. For multi class problems, the ensemble decision will be on the major class.

$$\mu_i^c(x) = \frac{1}{K} \sum_{k=1}^{K} d_{k,c}(x) \tag{4.6}$$

$\mu_i^c(x)$ is the *i*th expert giving the prediction on the class *c*. The commensurative based gate function chooses the class *j* that receives the largest total vote:

$$\mu(x) = \underset{j=1,\ldots,C}{\operatorname{argmax}} \sum_{i=1}^{K} \mu_i^j(x) \tag{4.7}$$

where $\mu(x)$ is the final decision of the gate function.

4.1.2.2 **Borda count based Gate Function**

The Borda count is a simple and effective method in combining the rankings of the experts. This method for ranked lists combination can be considered as a generalized model like the other proposed models [91]. In the Borda count based system, basically, the number of points given to the experts for each ranking is determined by the number of experts standing in the MoE system. Thus, under the simplest form of this method, where there are *N* experts, an expert will receive *N* points for a first preference, *N*−1 points for a second preference, *N*−2 for a third, and so on, with a candidate receiving 1 point for being ranked last (or left unranked). The Borda count for class *c* might be computed as

$$\mu^c(x) = \sum_{k=1}^{K} B_i^c(x) \tag{4.8}$$

where $B_i^c(x)$ is the ranking of the *c*th class provided by the *i*th expert and *K* is the number of experts. The final decision is given by selecting the class having the largest Borda count.

4.1.2.3 **Competitive Gate Function**

Only the winner expert in the network has the right to classify the query point *x*. This strategy is also called winner-take-all. In this style, the test observation is not classified by the contribution of the all experts in the system. Only the closest expert the test point makes the classification. For every input vector, the competitive experts compete with each other to determine which one of them is the closest to the particular query point. The gate function sets $w_i = 1$ as the output of the winner expert and the all weights of the other competitive experts will be set 0. w_i, is the *i*th expert weight in the *W* vector. The competitive gate function might be defined as:

$$\mu_j(x) = \min_{dist(x,E_j)\in R \text{ and } i\in R} \{\mu_i(x)\} \tag{4.9}$$

where *R* is the any real numbers. The closest expert to the test sample is the winner one. We would like to emphasize an important thing here assigning a class label to a query point by the only nearest expert might be problematic when the point is very close to a decision boundary. In this case, decisions of all the nearest experts should contribute to the calculation for accurate predictions.

4.1.2.4 **Cooperative (Collaborative) Gate Function**

In this method, the classification is done by means of the gate function giving different weights to the experts and can be named as weighted majority voting or distance weighted voting. Apart from the previous gate function models, all the experts have some weights on the calculation according to their distances to the query point. This method gives greater weights to closer experts. This effective approach is adapted to our study in designing the gate function. The formula (4.10) that finds the class probability of the class label *c* by the *K* experts is as follows:

$$\mu^c(x) = \sum_{i=1}^{K} w_i d_{i,c}(x) \tag{4.10}$$

where w_i is the weight assigned to the *i*th classifier.

The cooperative gate function gives greater weights to the closest experts by means of the Shepard's method [92]. Shepard's method is used in the design of gate function where closer experts to the test instance have higher effects and vice versa. It ought to be noticed that the closer experts according to their distances to the query point have higher effects, the further experts have lower effects in the calculation. Giving different weights to the experts by means of their inversely squared distance to the test point is done by this formula (4.11) below:

$$w_i = \frac{1}{dist_i^2} = \frac{1}{dist_i^2(x, E_j)} \tag{4.11}$$

where w_i is the weight of the ith expert. As a reference point, E_j is the centroid which represents the corresponding cluster (expert). There are some various types of weight methods such as $1/d$, $1/d^2$, and $1/d^3$. In contrast to the $1/d$ style, the $1/d^2$ style gives more weights to the closer experts and vice versa. In the experiments, the $1/d^2$ style produces better performances. Corporative gate function might be written as:

$$g_i = \frac{w_i}{\sum_{j=1}^{K} w_j} \tag{4.12}$$

where g_i is the weight of the ith expert. The total weight of the experts should be equal to 1 where *k* denotes both the number of the experts and the number of sub regions.

$$G = \sum_{i=1}^{k} g_i = 1 \tag{4.13}$$

The μ function is used to find the class of the new entry *x*, where $\mu: R \rightarrow C$.

$$\mu(x) = \underset{c \in C}{\operatorname{argmax}} \big(g_i . \mu_i^c(x) \big) \tag{4.14}$$

The μ function makes a calculation for each class and assigns the maximum argument as a class label. Here, *C* indicating the set of class labels is defined as $\{c_1, c_2, c_3, \dots c_s\}$. The $\mu_i^c(x)$ function calculates the probability of each class according to the ith expert. The extended style of the $\mu(x)$ function is in the formula (4.15):

$$\mu(x) = \underset{c \in C}{\operatorname{argmax}} \left(\frac{\sum_i^K w_i . \mu_i^c(x)}{\sum_j^K w_j} \right) \tag{4.15}$$

As it is seen, cooperative gate function gives different weights to the all experts.

4.2 EXPERIMENTAL RESULTS

In order to demonstrate the effectiveness and validity of the proposed methods, we have carried out some empirical studies which prove these proposed methods are comparably better than the former one. Experiments are performed on 36 UCI benchmark datasets [93] which represent a wide range of data characteristics, distributions and domains. In these real world datasets, missing values are replaced, nominal values are converted to binary values and finally all the values are normalized to [0-1] range. 5x2 cross validation technique are used throughout the experiments on the MATLAB environment in order to investigate the performance of class probability estimation techniques. The iteration number for *k*-means is 100, for all the experiments. Paired T-Test metric enables to compare the overall performances of the gate functions statistically [94]. T-Test gives three types of results: *win* (if the first classifier is better), *loss* (worse) and *tie* (equal). All computational results of the empirical studies are placed in Table 4. 1. The accuracy comparisons between the four types of proposed gate functions and the classical DT classifier are seen in this table. We would like to remind again that the DT is built using the entire dataset.

Table 4. 1 T-Test results of the whole gate functions

	k=3			*k*=5			*k*=10			*k*=20		
	# of *win*	# of *tie*	# of *loss*	# of *win*	# of *tie*	# of *loss*	# of *win*	# of *tie*	# of *loss*	# of *win*	# of *tie*	# of *loss*
Cooperative Gate F.	17	11	8	14	15	7	9	12	13	11	23	12
Competitive Gate F.	3	23	10	7	20	9	2	24	8	5	22	9
Commensurative Gate F.	2	10	24	1	8	27	4	5	27	2	4	30
Borda Count Gate F.	3	3	30	2	2	30	2	2	32	1	1	33

In each experiment, the number of experts (sub regions) has set to 3, 5, 10, and 20 and the accuracy comparison test is done with the DT against four kinds of MoEs. The numbers of *win*, *tie* and *loss* are presented in the table. As it is seen, these experimental results indicate that the cooperative gate function outperforms the others significantly in these evaluations. When the number of local experts is set to 3, this new method gives better results in 17 UCI datasets against DT. 17 *wins*, 11 *ties*, and only 8 *losses* indicate a good performance.

When the number of experts is set to 3, the all computational accuracy results between the DT and MoE are compared in Table 4. 2. In this table, the first three columns are the number of instances, attributes and the classes of each class respectively. These external features describe the density or sparseness of the future space. In the next two columns, there are accuracy rates of DT and MoE with the cooperative gate function. In the last two columns, MoE and DT are compered.

Table 4. 2 Experimental results when the number of experts is 3

	External Features					MoE - DT comparison	
Dataset Name	# of inst.	# of Att.	# of cls.	DT Acc.	Cooperative G.F. Acc.	% chang	T-Test
abalone	4153	19	10	0.2120	0.2203	3.92	*win*
anneal	890	4	62	0.9883	0.9357	-5.32	*loss*
audiology	169	5	69	0.8615	0.8166	-5.21	*loss*
autos	202	5	71	0.6614	0.6198	-6.29	*loss*
balance-scale	625	3	4	0.7799	0.7991	2.46	*win*
breast-cancer	286	2	38	0.6692	0.7098	6.07	*win*
breast-w	699	2	9	0.9465	0.9499	0.36	*tie*
col10	2019	10	7	0.7575	0.7738	2.15	*win*
colic	368	2	60	0.8092	0.8003	-1.10	*tie*
credit-a	690	2	42	0.8197	0.8400	2.48	*win*
credit-g	1000	2	59	0.6870	0.7011	2.05	*win*
d159	7182	2	32	0.9698	0.9572	-1.30	*tie*
diabetes	768	2	8	0.7036	0.7182	2.08	*win*
glass	205	5	9	0.6527	0.6429	-1.50	*tie*
heart-statlog	270	2	13	0.7415	0.7639	3.02	*win*
hepatitis	155	2	19	0.7883	0.8115	2.94	*win*
hypothyroid	3770	3	31	0.9943	0.9919	-0.24	*tie*
ionosphere	351	2	33	0.8539	0.8001	-6.30	*win*
iris	150	3	4	0.9333	0.9360	0.29	*tie*
kr-vs-kp	3196	2	39	0.9901	0.9790	-1.12	*tie*
labor	57	2	26	0.8596	0.8281	-3.66	*loss*
letter	2000	26	16	0.8234	0.8187	-0.57	*tie*
lymph	142	2	37	0.7901	0.7775	-1.59	*tie*
mushroom	8124	2	112	1.0000	0.9324	-6.76	*loss*
primary-tumor	302	11	23	0.4225	0.4348	2.91	*win*
ringnorm	7400	2	20	0.8842	0.9079	2.68	*win*
segment	2310	7	18	0.9396	0.9566	1.81	*win*
sick	3772	2	31	0.9824	0.9812	-0.12	*tie*
sonar	208	2	60	0.6989	0.7175	2.66	*win*
soybean	675	18	83	0.8824	0.8417	-4.61	*loss*
splice	3190	3	287	0.9254	0.8159	-11.83	*loss*
vehicle	846	4	18	0.6849	0.6787	-0.91	*tie*
vote	435	2	16	0.9361	0.9549	2.01	*win*
vowel	990	11	11	0.6773	0.6927	2.27	*win*
waveform	5000	3	40	0.7364	0.7512	2.01	*win*
zoo	84	4	16	0.9762	0.9333	-4.39	*loss*

Under the title of “% change” the values indicate the decrease or increase between MoE and DT in percentage. The values in the *% change* column show the percentage increase or decrease between the accuracy rates of Cooperative gate function and the accuracy rate of DT. The change in percentage is found by this formula:

$$\%\, change = \frac{Cooperative\ G.F.Acc. - DT\ Acc.}{DT\ Acc.} \times 100 \tag{4.16}$$

4.3 COMPLEXITY ANALYSIS

The general time complexity (cost) of a DT in big-O notation is as follows:

$$O(DN\log N) \tag{4.17}$$

where *D* and *N* are the number of dimensions (attributes) and instances in the data matrix respectively [90]. The complexity of the basic *k*-means method is also as:

$$O(DNIk) \tag{4.18}$$

where *I* and *k* are the iteration numbers and the cluster (region) numbers respectively [12]. The MoE mechanism comprises *k* DTs for each region and one gate function. Thus, the cost of the MoE construction is the combination of three factors: *k*-means, DTs, and gate function complexities:

$$O\left(DNIk + kD\frac{N}{k}\log\frac{N}{k} + k\right) \tag{4.19}$$

In this notation the first complexity part $(DNIk)$ is for *k*-means. The latter part $\left(kD\frac{N}{k}\log\frac{N}{k}\right)$ is the construction of *k* DTs where $\frac{N}{k}$ can be considered as the number of elements in each sub region. The last negligible part (k) is for the gate function which is almost the same for all types. The constant *k* parameter might be cancelled as it is a small integer in most of the applications. Because of this, the complexity of *k* DTs is not as big as it is expected. In addition, the iteration number *I* can also be negligible since it is a constant value throughout the experiments. If the values *I* and *k* are cancelled, the notation might be approximately summarized as:

$$O(DN\log\mathrm{N}) \tag{4.20}$$

Even though the cost of MoE construction is the same as DT's, the computational time should be a little bigger.

As another complexity measure of DT learners, the number of nodes in the tree is an important parameter. The number of nodes in the DT classifier and MoE classifier is listed in Table 4. 3. The total number of nodes of the experts in the MoE system is calculated when the number of expert is set to 5. As it is seen in the table, there is big similarity. This table proofs that the complexities of both models are similar to each other.

Table 4. 3 Number of nodes in the DTs and MoEs

ID	Dataset	Number of Nodes in DT	Number of average nodes in MoE (k=5)
1	abalone	1515	1502
2	anneal	21	34
3	audiology	37	42
4	autos	51	58
5	balance-scale	69	74
6	breast-cancer	63	58
7	breast-w	37	22
8	col10	295	396
9	colic	47	50
10	credit-a	93	106
11	credit-g	209	198
12	d159	205	462
13	diabetes	139	138
14	glass	47	66
15	heart-statlog	35	66
16	hepatitis	19	30
17	hypothyroid	23	78
18	ionosphere	37	46
19	iris	9	20
20	kr-vs-kp	73	202
21	labor	9	12
22	letter	2413	2600
23	lymph	21	28
24	mushroom	27	48
25	primary-tumor	81	70
26	ringnorm	531	600
27	segment	91	130
28	sick	53	76
29	sonar	35	44
30	soybean	81	120
31	splice	167	220
32	vehicle	151	170
33	vote	19	26
34	vowel	191	240
35	waveform	737	620
36	zoo	7	10

4.4 EVALUATION AND FUTURE STUDIES

In this study, the feature space is clustered into disjoint regions and a special classifier is established for those regions. An assigned gate function integrates the decisions of the classifiers. It has been seen that a locally adaptive models gives more accurate results as opposed to a generalized single model.

In the DT learning, the data points are split into two parts according to the Information Gain (IG). Nevertheless, in clustering phase of this study, the data points are divided into sub regions according to the statical data distributions. This approach gives a great benefit when the dataset is linearly separable (e.g., *balance-scale*, *heart-statlog* and *waveform*). While this proposed method outperforms on some datasets, it also fails on few datasets due to some reasons. Basically the accuracy comparisons shown in Table 4.1 and Table 4.2 are directly related with the suggested gate functions. The clustering method also considerably has a great role in the performance. The statistical type, the distributional style [95] and some related meta-features [81] of the dataset have a great importance on the clustering process. The discretized regions sometimes cannot be suitable for the proposed MoE model. In a future study, the appropriate types of datasets can be categorized.

4.4.1 Clustering method

In the beginning of the study, we have assumed that it was possible to compare the performances of two different clustering algorithms EM and k-means. Nevertheless, EM is not applicable to some datasets. Where *k*-means can be run over any datasets, EM can run only a few datasets. EM has produced some ill-conditioned situations on some real world UCI datasets. Ill-conditioned covariance occurs, when there is an error in fitting the Gaussian Mixture Model (GMM) [96]. Afterwards, optimization ends and an error appears during the implementation. In this situation, we have tried to fit a GMM again using regularization. Although we set a small positive scalar value to Regularization value (e.g. 0.01) in order to ensure that the estimated covariance matrices positive definite, the ill-conditioned covariance could not be disappeared.

The MoE using EM is not applicable to any type of dataset. On the other hand, the *k*-means method does not care the data distributions in the dataset. Therefore, this scalability, flexibility and applicability of the k-means method provide a great privilege according to the GMM.

4.4.2 **Gate Functions**

Proposed gate functions perform the classification phase by giving some weights to the local experts. When the performances of the gate functions are ranked, Cooperative gate function model comparably gives the best results of all. Commensurative and Competitive ones are not as good as Cooperative one. On the other hand, the Borda count model is the worst one as shown in Table 4. 1.

In the cooperative version, each expert has some contributions on the classification the test point. Especially the closer experts to the test point have bigger effects to the calculations due the Shepard method, namely the $1/d^2$ effect. Thus, the distance based model enables accurate prediction.

The competitive model which uses only one expert has some drawbacks. For example, a point located closely to a decision boundary or a class boundary consults to the nearest expert where there might be some other experts nearby. This situation might cause misclassification by assigning a class label to the observation by the help of only the nearest expert. In this case, the cooperative gate function solves this kind of problem by contributing the other experts.

The Commensurative function assigns equal weights to each of the experts. Since the weights of the closer and the farthest experts are identical, misclassification usually happens. Hence, the overall performance of this version is not better than DT’s.

The Borda count method normally is one of the useful and effective methods in some election and decision making processes. However, as a gating function, this method fails in this application. This method actually is a kind of commensurative version of gate function. Giving equal weights to the farthest experts makes the classification phase unsuccessful. As a further empirical study, Borda count method using Shepard’s technique might give more accurate results than the cooperative one.

4.4.3 *k* Dependency

In this study, this improved system requires a *k* value defined by the user's preference. It is mostly difficult to determine the best *k* value. In the literature it is commonly recommended to assign the best *k* value for a dataset by carrying out some experiments. The K fold cross validation process is the first simplest choice. Using some kind of validation process for different *k* values, the best one that gives the highest accuracy might be selected. The dependency of parameter is a common problem in every learning process. But there are some researches such as [95] and [97] that find the most suitable number of clusters (*k* parameter) for a dataset. Also in this paper [98], automatic number of cluster detection is improved using Kohonen's Self Organized Map (SOM) [99].

In the experiment, four types of gate functions with the different number of *k* values (3, 5, 10, and 20) are applied to the UCI datasets. If the big *k* value (# of experts) is chosen as a parameter, the execution time gets bigger. Additionally, dividing sparse datasets into clusters gets difficult. Therefore, increasing the *k* value towards to the number of samples in the dataset causes a situation like a *distance weighted k*-NN structure when the cooperative gate function is used.

In the experiments, it can be said that assigning more *k* experts to the dense datasets and vice versa less *k* experts to the sparse datasets gives more accurate results. On the assumption the density of a dataset might be the frequencies of the data points in the feature space; we can use this formula (4.21) below that describes the denseness of a dataset [58]:

$$Density = \frac{N}{A \times C} \tag{4.21}$$

where *N*, *A* and *C* are the number of data points, attributes and class labels respectively. In Table 4.4, increasing the number of experts in the dense dataset gives higher accuracy increments. However, in the sparse datasets as shown in Table 4.5, bigger number of experts gives lower accuracy decrements. The accuracy comparisons in both Table 4.4 and Table 4.5 are done between DT and MoE using cooperative gate function.

Table 4. 4 Accuracy in dense datasets

					% change in Acc. using Cooperative G.F.			
	N	*A*	*C*	*Density*	*k*=3	*k*=5	*k*=10	*k*=20
balance-scale	625	3	4	52.1	2.46	3.77	5.29	5.53
breast-w	699	2	9	38.8	0.36	1.36	1.33	1.48
iris	150	3	4	12.5	0.29	0.58	1.15	2.00
ringnorm	7400	2	20	185	2.68	4.84	4.82	6.80
waveform	5000	3	40	41.7	2.01	0.44	3.24	6.31

In some UCI datasets shown in Table 4.4, increasing the number of experts gives more accuracy rates. This situation has some drawbacks. The first one is the high computational time. The second one, when the *k* parameter is chosen a huge number, sparse sub regions occurs where DT classifiers act like distance weighted *k*-NN classifiers. Namely, when the *k* value approaches to the number of samples, a model like $1/d^2$-distance-weighted-*k*NN classifier occurs. On the other hand, as it is seen in Table 4.5, there is no need for the sparse dataset to use the proposed model. This MoE model decreases the accuracy rate.

Table 4. 5 Accuracy in sparse datasets

					% change in Acc. using Cooperative G.F.			
	N	*A*	*C*	*Density*	*k*=3	*k*=5	*k*=10	*k*=20
audiology	169	5	69	0	-5.21	-13.46	-21.56	-40.11
autos	202	5	71	0.2	-6.29	-11.68	-21.11	-30.84
colic	368	2	60	0	-1.10	-0.53	-5.17	-9.53
labor	57	2	76	0.5	-3.66	-11.42	-11.83	-9.38
soybean	675	18	83	0.5	-4.61	-9.91	-15.92	-17.26

4.4.4 **Future Studies**

As a future study, the performance comparison between the MoE with hard clustering and the MoE with soft clustering might be enabled by generating some artificial datasets. The produced synthetic datasets enables the discretization by any type of clustering algorithms such as EM, DBSCAN and k-means. Then, the performance comparison among k-means, EM and DBSCAN will show the performance differences of models.

In the further steps, rather than DTs, other classifiers including generalized or linear regression models can be applied to the proposed method for the purpose of better performance.

Besides, an automated system that finds the best k parameter fitting to a dataset might be established. Since the searching process of finding the best k value (the number of experts) is very time-consuming, it limits its use in many real-world applications. In the recent years, several approaches depending on the data distribution types have also been suggested and studied in order to tackle these mentioned problems.

After having the computational results, the reasons of the differences in the performance on some datasets are thought to be examined. Although the proposed MoE model with the cooperative gate function is very successful on some datasets, the reasons can be analyzed through the Meta Learning process. Some meta-features of the datasets such as the external features and geometrical complexity measures proposed by [81] might be used in this learning type.

The extracted features and the T-Test results of the MoE system $(loss = -1, tie = 0, win = +1\ )$ might be accepted as attributes and class labels respectively in a Meta Learning dataset. By using a regression model the T-Test result of a dataset with a certain k parameter can be predicted. Amasyali and Ersoy [100] have made a thorough research that divides the Meta Learning activity into categories. According to the technical report study, different types of used meta-features are categorized into these classes: statistical, informational and theoretical features, subsampling landmarks features, and DT features. About 300 meta-features are aggregated into their study and used in Meta Regression model.

In the cooperative version, the farther experts to the test point have smaller contributions to the calculations due the Shepard method, namely the $1/d^2$ effect. On the contrary, including the decisions of the farther experts to the gate function increases the execution time. Consequently, a mechanism can be established that only the nearest related experts among the test point can play a role in the calculation as a further study.

4.5 **SUMMARY**

In this theoretical and practical study, a dataset is divided into sub partitions by a hard clustering method. An expert for each sub region is assigned and trained with the points of the corresponding region. The decisions of the experts are merged by four types of gate functions: cooperating, competitive, commensurative, and Borda count. In the experiments, better performances have been obtained mostly on dense datasets with the proposed cooperating gate function due to its mechanism that gives different weights to all of the experts. Normally, MoE model is particularly useful when different types of experts are trained on different parts of the same feature space. Training a set of chosen experts and assigning the best ones to each division might be another expensive choice in the training phase. But in this study the same type of experts are chosen rather than heterogeneous ones throughout the experiments. It is also showed that the total classification performance might be increased with this suggested MoE approach using a cooperative gate function and homogeneous experts on hard clustered regions. In addition, this study enables the suggested MoE model to be applicable to any type of dataset unlike the other models like GMM.

4.6 **AVAILABILITY**

The implemented MATLAB codes of the proposed model with four types of gate functions (cooperating, competitive, commensurative, and Borda count), the overall experimental results in pure text files and Excel documents and related files can be publicly downloaded from the URL address for examinations and further studies: https://www.ce.yildiz.edu.tr/personal/mfatih/file/5072/moe.rar

CHAPTER 5

A LOCALLY ADAPTIVE BASE LEARNER: ONE NEAREST CLUSTER

Classifiers are arranged into two main categories, *eager* and *lazy*. In contrast to *lazy* methods (e.g., *k*-NN, PART and One-Rule) *eager* ones (e.g., Decision Trees, SVM and MLP) builds a generalized model from the training set. Basic *lazy* methods search the entire dataset for each test instance. k Nearest Neighbors (k-NN) classifier is one of the most preferred algorithm in this area. The *k*-NN assigns the class label which is most frequent among the *k* training samples nearest to the query point. In other words, the test sample is classified into a particular class by the majority voting of the *k* closest training samples. Because of this, the closest training examples have a considerable influence on the classification accuracy. This memory based classification algorithm is used with a constant *k* value defined by the user's preference. It is generally difficult to determine the best *k* value. In the literature it is commonly recommended to assign the best *k* value for a dataset by carrying out some experiments [101]. A constant *k* value for each test instance may results low accuracy rates.

This study aims to improve a sample based base learner by using unsupervised methods. Our research which differs from some proposed models is about a novel way of assigning a *k* value best fitting to each test instance to take better performance. The proposed research aims contributing a dynamic *k* parameter selector to the traditional *k* Nearest Neighbors algorithm. The proposed hybrid method combines supervised and unsupervised techniques. This study also aims to remove the side effects by proposing a novel method, dynamic *k* value selector for each test instance. To show the side effects of the constant *k* value, five evidences are presented below.

5.1 The 1st evidence

As it is seen in Figure 5.1, in Figure 5.2, in Figure 5.3, and in Figure 5.4, there are accuracy rates of the *k*-NN classifiers with different *k* parameters on some UCI benchmark datasets [93]. All accuracy values are obtained using 10 fold cross validation throughout the experiments. The accuracy rates have been computed according to the constant *k* values ranging from 1 to 50 along the classification activity. Whereas the *k* parameter changes, the accuracy level obviously changes as well.

As it is seen in Figure 5.1, increasing the *k* value decreases the classification performance of the datasets *vovel, ionosphere, soyabean, and segment*. Especially, the *vovel* dataset is very sensitive to the high *k* value.

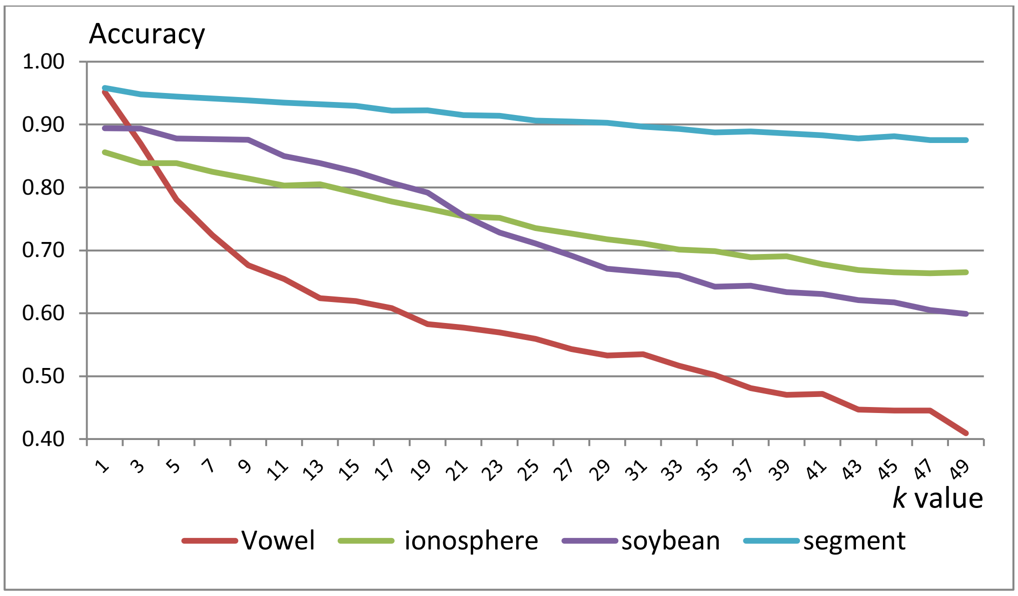

Figure 5.1 The negative effect of increasing *k* parameter

On the other hand in Figure 5.2, high accuracy level is directly related with the high *k* value. As it is seen in the *balance-scale, waveform, splice, and heart-statlog* datasets, increasing the *k* value gives higher accuracy rate.

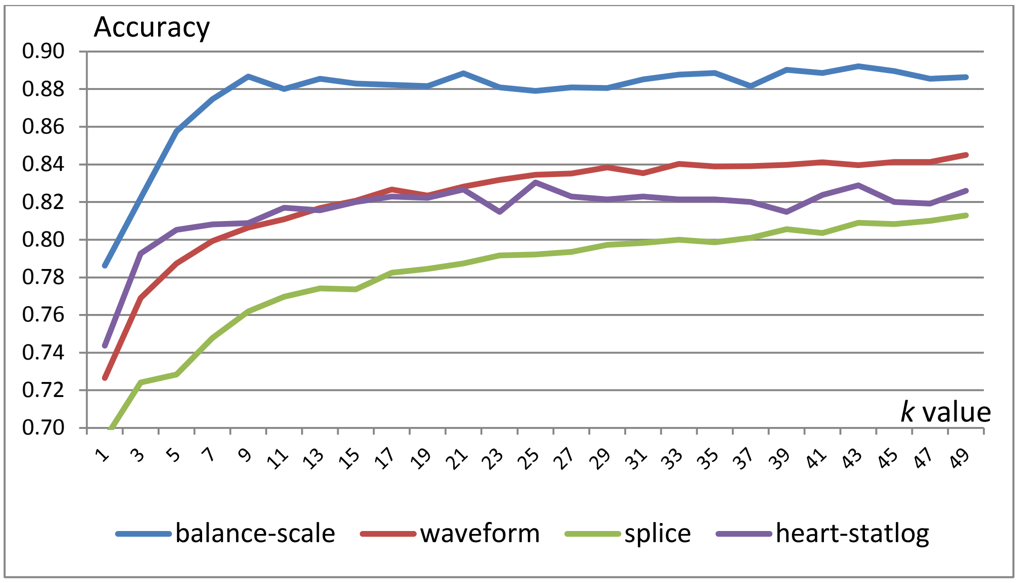

Figure 5.2 The positive effect of increasing *k* parameter

Besides, In Figure 5.3, the datasets *iris, hypothroid, kr-vs-kp,* and *sick* are not sensitive to the *k* parameter. There are gradual changes in the performance. There are only some slight changes on account of the *k* parameter.

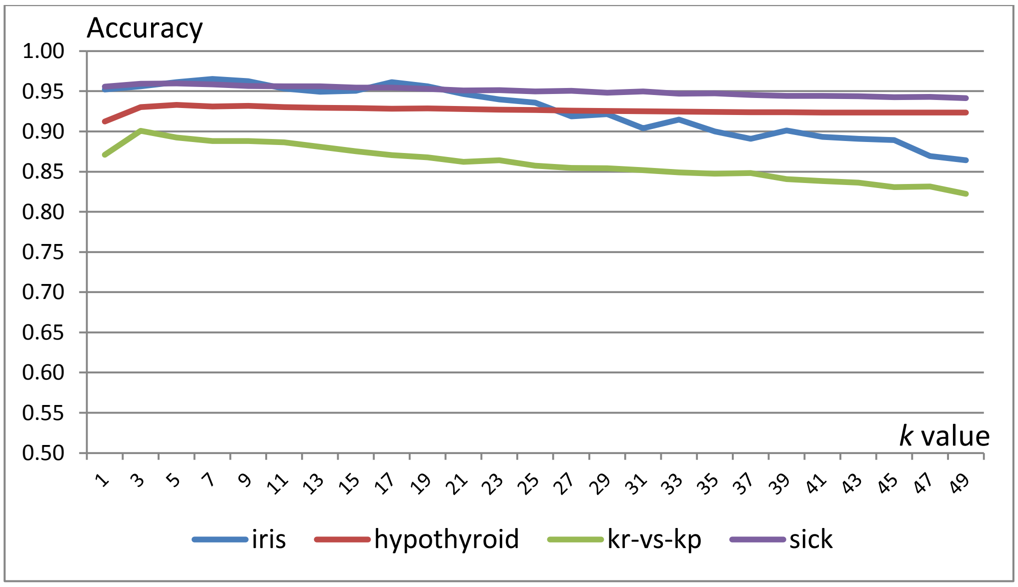

Figure 5.3 The effect of increasing *k* parameter

Nevertheless, Figure 5.4 represents a different situation. There is not a linear relation between the *k* value and accuracy level. The *k* value sometimes has a positive correlation effect. But it sometimes has a negative correlation effect on *colic*, *credit-a,* and *sonar* datasets.

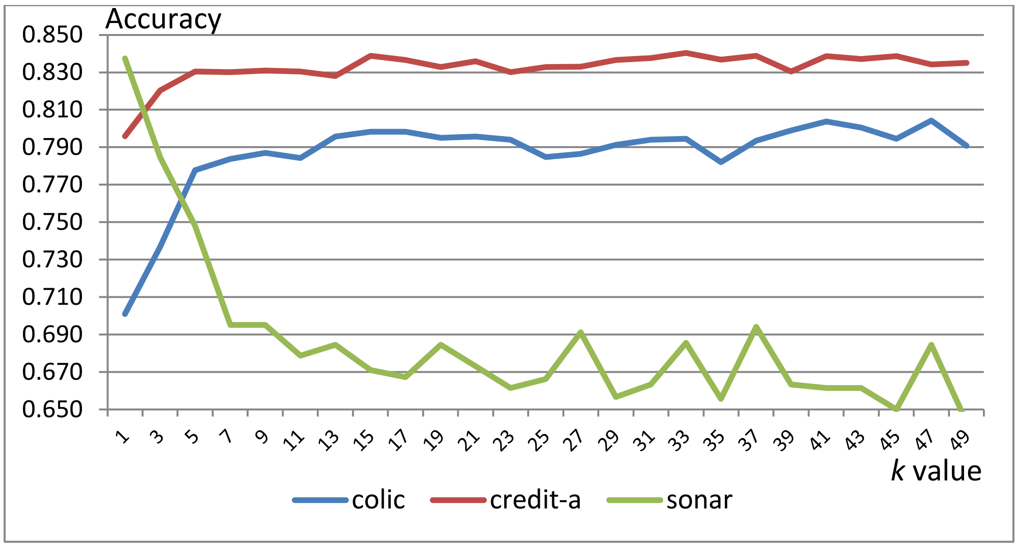

Figure 5.4 The effect of increasing *k* parameter

In Figure 5.5, there is a summarization of the first evidence described in the figures above. The *k* parameter has four types of effect on the accuracy level. These are no correlation (as it is seen in the *sick* dataset), positive correlation (as it is seen in the *splice* dataset), negative correlation (as it is seen in the *ionosphere* dataset), and finally changeable correlation (as it is seen in the *audiology* dataset). As a result, this figure depicts the classification and regression performance of the *k*-NN method may be very sensitive to the *k* parameter.

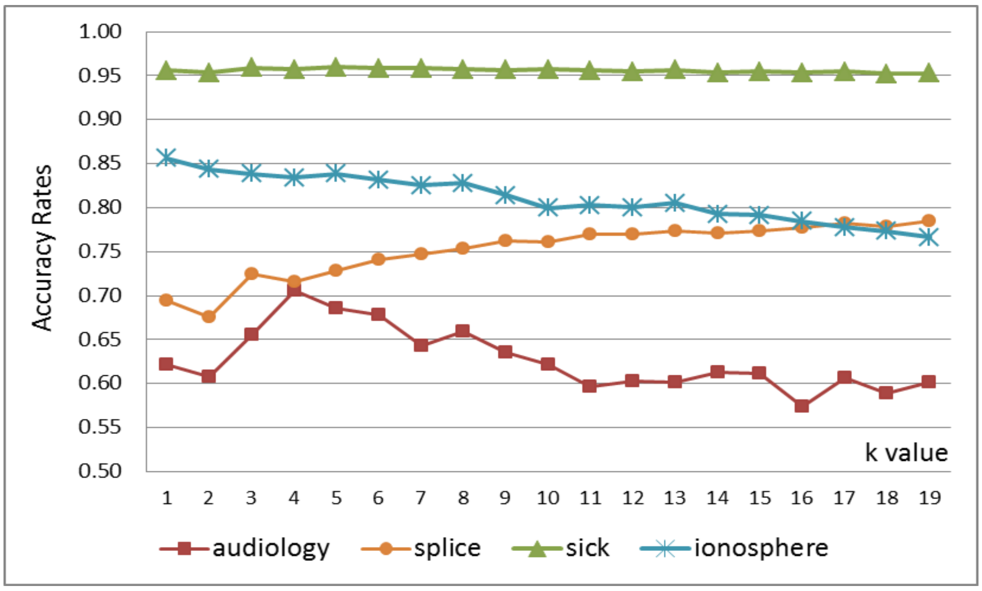

Figure 5.5 Effects of the k parameter on k-NN performance

5.2 The 2nd evidence

There is no unique *k* parameter that gives the highest cumulative accuracy rate and Table 5. 1 proves this. The table presents whether the different *k* values for randomly selected test points in the *audiology* dataset gives accurate classification or not. T and F represent the true and false predictions respectively. T indicates that the classifier predicts the test sample's class correctly and vice versa. The italic real numbers under the T and F letters indicate the probability of the prediction in percentage. An object is classified by a majority vote of its *k* closest neighbors. For example, the class probabilities of the 33rd test point are almost 1. This point should be in the center of a spherical group of points from the same class. The 61st test point is correctly classified when the *k* is bigger than 2. The estimation of the 70th test point is unstable thanks to the different *k* values. It might be because the test point is near a decision boundary. This table apparently confirms that the classification performance is very sensitive to the precise value of *k*.

Table 5. 1 Analysis of the k-values effecting performance

	k parameter for *k*-NN									
	1	2	3	4	5	6	7	8	9	10
33rd test point	T *1.0*	T *1.0*	T *1.0*	T *1.0*	T *1.0*	T *1.0*	T *1.0*	T *1.0*	T *0.88*	T *0.9*
54th test point	F *1.0*	F *0.5*	F *0.66*	F *0.75*	F *0.80*	F *0.83*	F *0.85*	F *0.75*	F *0.66*	F *0.60*
61st test point	F *1.0*	F *0.5*	T *1.0*	T *0.75*	T *0.6*	T *0.83*	T *0.71*	T *0.75*	T *0.88*	T *0.9*
70th test point	F *1.0*	F *1.0*	F *1.0*	T *0.75*	T *0.6*	T *0.66*	F *0.71*	F *0.62*	F *0.67*	T *0.6*
87th test point	F *1.0*	F *1.0*	F *0.66*	T *0.75*	F *0.6*	F *0.66*	F *0.71*	F *0.75*	F *0.78*	F *0.7*
# of true prediction	**1**	**1**	**2**	**4**	**3**	**3**	**2**	**2**	**2**	**3**

According to the table, when the *k* parameter is fixed to 1, only 1 class label of the test samples is truly predicted. If the *k* value is set to 4, 4 test points in total are accurately classified. Moreover, different *k* values suited for each test point increases the overall accuracy as opposed to constant *k* values. The value of *k* for each test point should be small, big or in a specific interval for the purpose of higher accuracy.

5.3 **The 3rd evidence**

In Figure 5.8, Figure 5.7 and Figure 5.8 there are histogram plots of the *zoo*, *breast-cancer* and *hepatitis* datasets respectively from the UCI repository. In these datasets, all the values are normalized to 0-1 range. All the distances between the origin and all points have been calculated. Then, the points are laid on one dimensional axis according to their distances to the origin to build the histogram plots. Namely, each histogram displays the frequencies of distances between the origin and all points in the space. Each histogram gives a notion about the data distribution among the origin. The histogram plots absolutely change if the test point is moved to another location of the space.

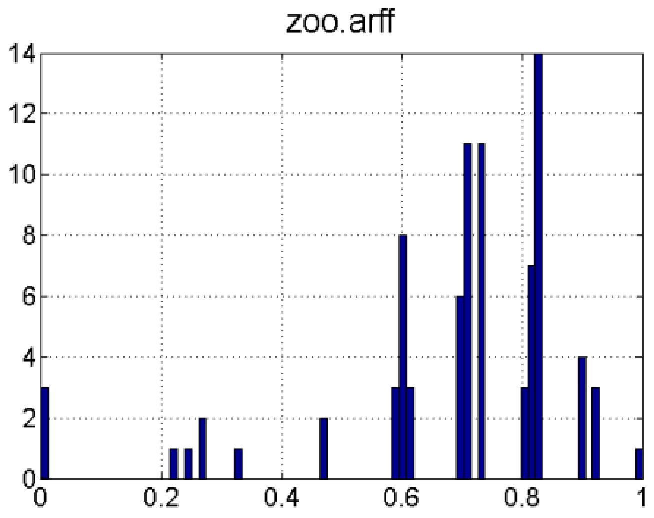

Figure 5.6 Histogram of the *zoo* dataset

Assuming a test point in the origin of the *zoo* dataset, as it is shown in the histogram plot above, the best *k* value should be set to 3 since there are three nearest points in the same orbit. These three points are considered together due to the same distance even though they are very far each other in their original space. Because of this, these three points are called in the same orbit.

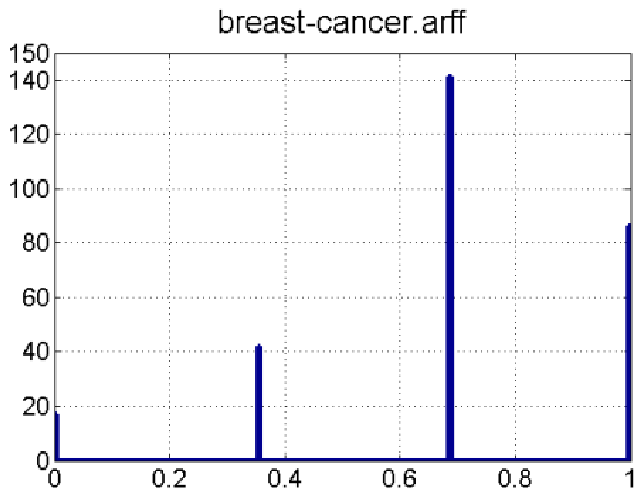

Figure 5.7 Histogram of the *breast-cancer* dataset

The best *k* value for a test point in the origin of the *breast-cancer* should be approximately 15 according to the histogram plots above. In the *hepatitis* dataset, the *k* value should be 1. These histograms prove that there should be a suitable *k* value for each test point due to the distribution of its nearest neighbors on one axis.

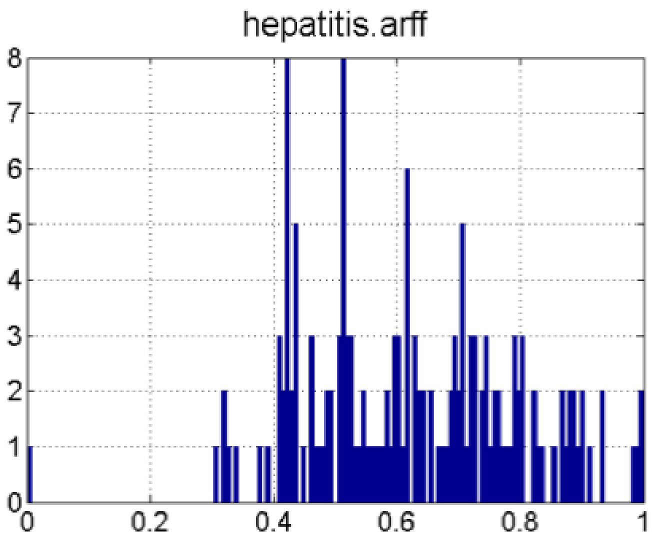

Figure 5.8 Histogram of the *hepatitis* dataset

There are three histogram plots of the same dataset *labor* in Figure 5.9, Figure 5.10 and Figure 5.11. These figures show the number of points among the origin, the center and the upper rightmost point of the space respectively. The histogram plots of the test instances are different from each other although the space is the same. Instead of setting the *k* values for each test point to 1 in *k*-NN classifier, they should be 2, 3 and 1 respectively to have reliable and accurate results. In these plots, the numbers of nearest points to the origin, center, and the right upper most corner are circled for illustration.

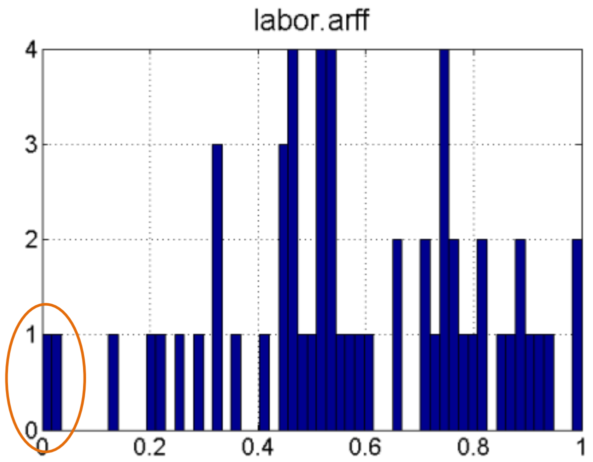

Figure 5.9 Histogram of the origin point in the *labor* dataset

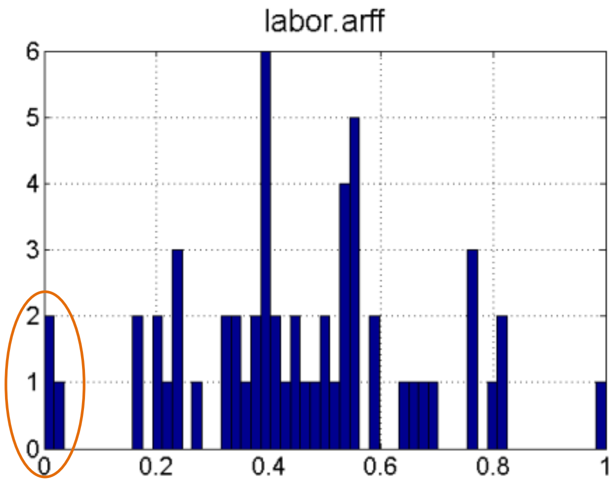

Figure 5.10 Histogram of the center point in the *labor* dataset

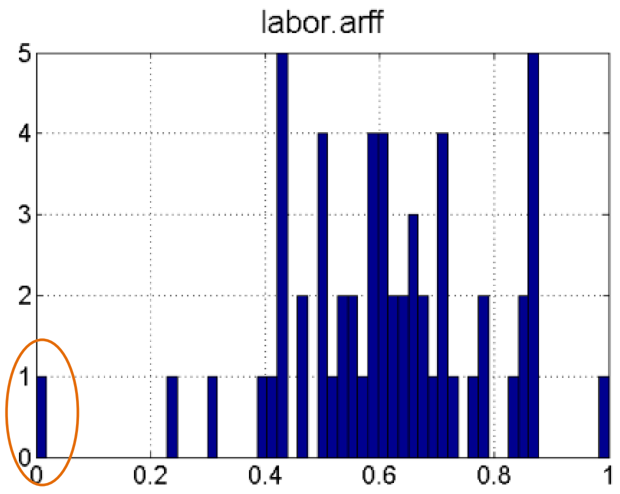

Figure 5.11 Histogram of the right upper most corner in the *labor* dataset

On the contrary, the *letter* dataset has a different data distribution shown in the figure below. In this dense dataset, the best *k* parameter for the test point located in the origin might be found by experiments and cross validation procedures. In this case, handling the whole dataset is not a good choice, selecting some nearest samples to the test point and examining the distributions is better. Figure 5.13 and Figure 5.14 show the closest 100 and 20 data points respectively to the origin. According to Figure 5.14 the best *k* value for the test point might be 1.

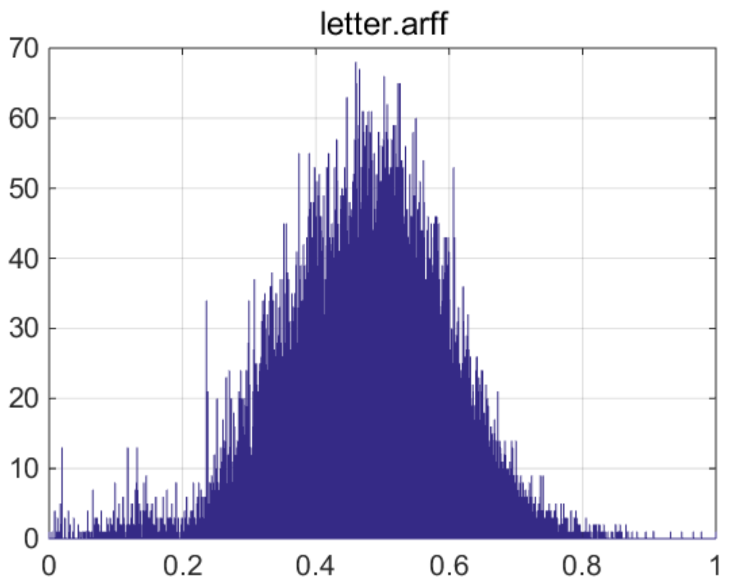

Figure 5.12 Histogram of the *letter* dataset

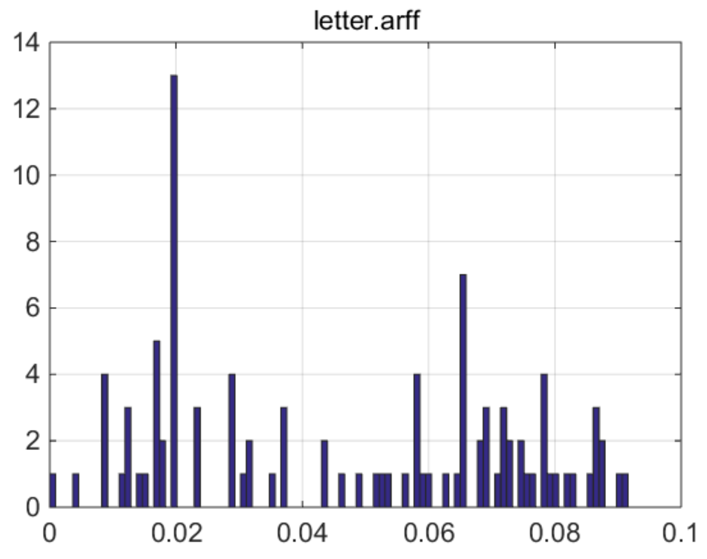

Figure 5.13 Histogram of the nearest 100 points in the *letter* dataset

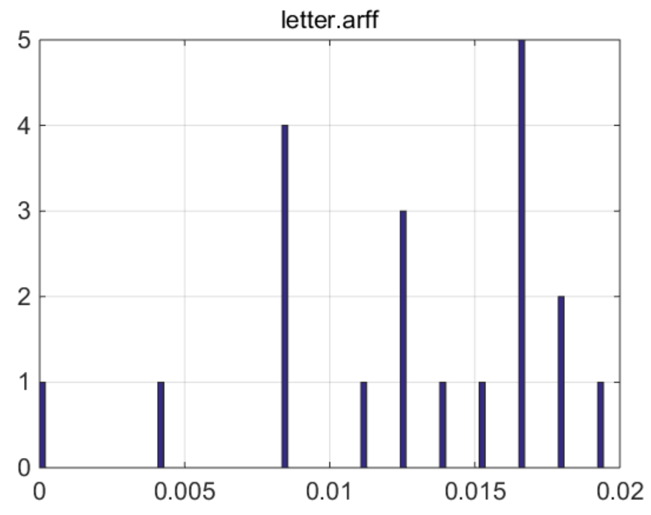

Figure 5.14 Histogram of the nearest 20 points in the *letter* dataset

5.4 **The 4th evidence**

There are two different scenarios in two dimensional space illustrated in Figure 5.15. The scenarios in the (a) section demonstrate that there might be other samples whose distances are similar to the test point. In this case, setting the *k* value to 1 results in unreliable classification because of the random selection. Normally, the probability of having the same distance to the test point is very low. Conversely, a group of samples whose distances are similar to the test point might occur as it is seen in the (b) section. These points might also be very close to each other. To assign a value which is bigger than 1 to the *k* yields different results. Instead of this, setting the *k* value to the number of the samples in the nearest group produces more reliable classification.

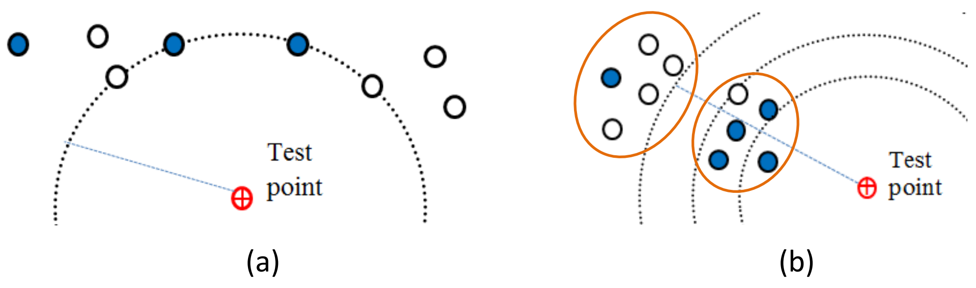

Figure 5.15 Classification example

5.5 The 5th evidence

Figure 5.16 shows the intervals of the best *k* values giving the highest accuracies for the datasets on the *y* axis. For example, *k*-NN gives the highest accuracies on the *zoo* dataset when the k value is between 1 and 9. In other words, the accuracy rates are similar between [1, 9]. The intervals are computed when the standard deviation of the accuracy rates is set to 0.01.

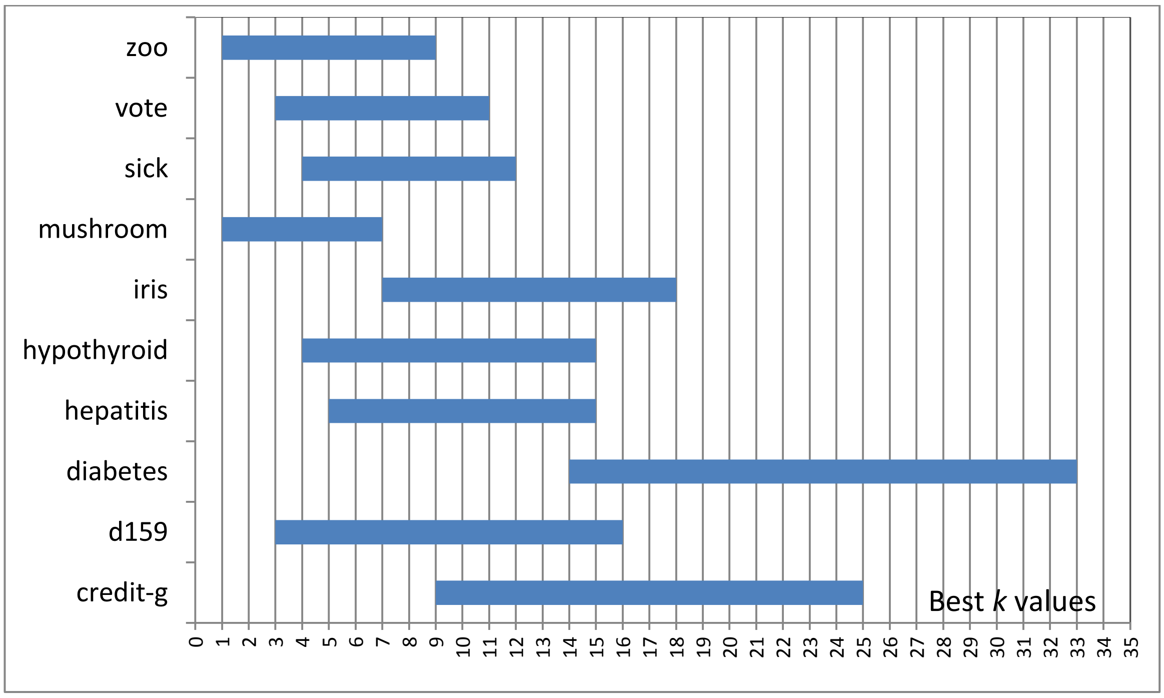

Figure 5.16 The intervals of the best *k* values giving the highest accuracies for some datasets

This figure shows that there is not a unique *k* value that gives the highest classification performance. Thus, the k value for some datasets might be laid in some intervals.

5.6 ***k* Dependency**

The performance of the *k*-NN classifier is strictly related to the *k* value. *k*-NN with small *k* values are very sensitive to the noise data. In order to reduce the effect of noise, the *k* value can be slightly increased. Nonetheless, if the *k* value exceeds its convenient level, it includes unnecessary points from other classes and results unreliable predictions. This signifies that high *k* value is prone to misclassification. Additionally, attaining better accuracy with adequately high value of k requires high computational cost, because of the complexity of NN searching. As it was mentioned above, there is no exact rule and proper method of *k* estimation. It depends on the features of the input space such as data distribution type, structure, density of data, the number class labels, the separability of the class boundaries, and some other meta-features. In different scenarios, the optimum *k* might change. The value of optimum *k* totally depends on the dataset. In addition, small, moderate or big *k* values generally yields different results. Hence, the *k* value may vary from dataset to dataset. The value of *k* is extremely training-dataset dependent.

In addition, each section in the same dataset might be statistically different from the others in various aspects. These differences can be related with the number of samples, the number of classes, and the number of attributes. All of these features are the determinant factors describing the sparseness and denseness of the dataset. Additionally there might be some other geometrical meta-features [81] affecting the performance of the classifier. In order to boost the overall performance of a *k*-NN classifier, it will be better to build a mechanism that assigns a locally adaptive *k* parameter to each instance.

5.7 **Drawbacks of *k*-NN**

Lazy learners have expensive computational costs since they store all the training instances and do not build a generalized model. *k*-NN has two main drawbacks. The first one is the high estimation time because of the absence of a generalized model [102]. The second one is the *k* dependency which controls the volume of the neighborhood.

In recent years, researchers have done considerable studies on the various types of *k*-NN classifier such as combining it with other techniques and using different distance metrics. Several approaches have also been suggested and studied in order to tackle these mentioned drawbacks. Jiang at al [103] presented three shortcomings of *k*-NN in their research:

1. Type of the distance metric for measuring the difference or similarity between two samples.
2. Artificially assigned neighborhood size as an input parameter.
3. The class probability estimation with a simple voting technique.

They propose some solutions and methods to these shortcomings and then have better performances in their experiments.

In the latest surveys of *k*-NN method [102] and [104], advantages and disadvantages of the several versions of this method are examined. High memory requirement and computational complexity are presented as two main drawbacks of *k*-NN. In order to improve over memory limitations, training dataset are structured using some algorithms such as ball tree, kD-tree, nearest feature line (NFL) [105], tunable metric [106], principal axis search tree [107] and orthogonal search tree [108]. These methods simply decrease the computational time of NN algorithms.

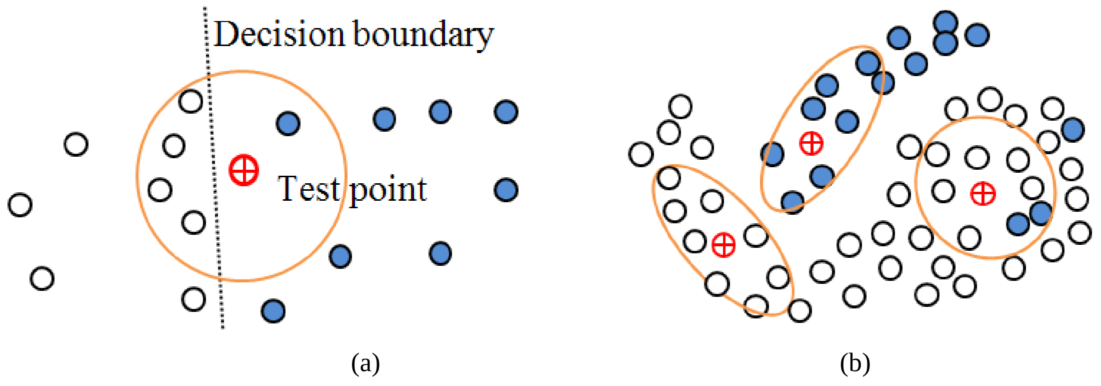

Figure 5.17 Illustrations on misclassification

We have determined that two cases where *k*-NN classifier fails in the experiments. These two cases are illustrated in the Figure 5.17. Firstly, when the *k* value is high, *k*-NN frequently fails while labeling the test samples which are closely located to the decision boundaries. Thus, the class boundaries will not be precise any more. In Figure

5a, the vertical line represents the class boundary between white and blue classes in the two dimensional dataset. The decision boundary is represented by the black dotted vertical line. The red query point, in which there is a plus sign, is very close to the decision boundary. It will be misclassified since the *k* value has been set to 5 as illustrated by the orange circle in the figure. Although the test point is in the region of blue dots, its class will be white because of the higher *k* value. *k*-NN fails here for that reason.

In Figure 5.17 (b) showing the second case, there are different neighborhood shapes where the orange ellipses and circle comprise the adaptive 8 nearest points, (8NN). The shape varies with the location of the query instance thanks to the distribution of its nearest samples [109]. Using standard unweighted Euclidean distance metric causes misclassification in these elliptic regions. The amount of elongation/restriction decays as the query point moves further away from the elliptic regions where a decision boundary would lie. The elliptic illustrations require locally adapting a distance metric. There are some researches [109], [110] using adaptive and discrimination metrics in order to boost the performance of nearest neighbor classifiers in these two cases illustrated in Figure 5.17.

Patterns of different classes sometimes overlap in some regions in the space. Instances of different classes can be very close to the decision boundaries. In these circumstances, *k*-NN includes some irrelevant instances to the calculation. Numerous researchers have proposed various adaptive and discrimination metrics [110] and [109] to improve the performance. Subsequently, the adaptive *k*-NN rule with the convenient distance measure might be used to overcome these hardships for further studies.

As mentioned above it is explicitly proved that a new method should be built to find the best fitting *k* value for each test point so as to boost the prediction performance of the *k*-NN classifier. There are also some explanations of the reasons why *k*-NN fails on some datasets. The rest of this chapter has some more sub-sections. In next section, there are briefly described studies in the related surveys and researches. In the following section, there is a definition about the dynamic *k* parameter selection procedure and search techniques used in the instance based learning. Experimental results are presented in the following section. Finally, future work is presented and our contributions are summarized.

5.8 RELATED STUDIES

There are two main empirical evaluation approaches about setting the appropriate *k* value for a particular dataset. The K fold cross validation process is the first simplest choice. Using some kind of validation process for different *k* values, the best one that gives the highest accuracy might be selected. The widely used second approach is the bootstrapping technique. It uses sampling with replacement to form the training set [111]. The optimal *k* value is determined via bootstrap method. Both approaches give approximately the same results. In spite of some suggestions [19] about setting the *k* value to the square root of the number of all training patterns, it is theoretically an upper bound value that cutbacks these kinds of evaluations.

Ozger et al [20] proposed an approach assigning the appropriate *k* value for a particular dataset by means of Meta Learning method. In their study, 16 meta-features are extracted from each of 200 datasets. The *k*-NN algorithms with different *k* values are computed with these datasets. In the construction of Meta training dataset, the *k* value giving the highest accuracy becomes the label and the extracted meta-features are accepted as attributes. It predicts the best fitting *k* value by a regression model. Nonetheless, the biggest barrier in front of the study is the assignment of the same *k* value to the whole datasets. The highest accuracy for more than half of the datasets is computed where the *k* parameter is 1. For that reason, regression becomes difficult.

In another research [21], some non-parametric *k*-NN where the general *k* for a dataset is automatically determined by geometric relationships is proposed. Classification is done through the centroids which globally represent the classes. Increasing the *k*-NN performance is obtained by the estimation of the optimal *k* parameter or making the *k*-NN algorithm adaptive to data by means of determining local decision boundaries.

Ghosh [70], and Guo et al. [23] have proposed some techniques that finds a globally adaptive *k* value for a dataset. On the other hand, in another research [22], Ghosh has presented a locally adaptive nearest neighbor classification technique, where the value of *k* is automatically selected depending on the distribution of competing classes in the vicinity of the test point to be classified. The distribution of the nearest samples plays a great role in this technique.

Our research which differs from these studies mentioned above is about a novel way of assigning a *k* value best fitting to each test instance to acquire better performance. The proposed research is aimed at contributing a dynamic *k* parameter selector to the traditional *k*-NN algorithm. The proposed hybrid method combines supervised and unsupervised techniques.

5.9 CLASSIFICATION WITH ONE NEAREST CLUSTER (1NC)

All the situations and scenarios mentioned in the beginning of this chapter section indicate that there should be a different and suitable *k* parameter for each test point in the *k*-NN classification activity. The proposed hybrid algorithm which gives more accurate results is detailed in this section. The steps of the algorithm, named as *One Nearest Cluster* (1NC), are as follows:

1. Choose *M*, *l* (# of clusters), and *I* (# of iteration) parameters.
2. Take *M* closest samples around the test sample.
3. Lay these closest samples on one dimensional axis according to their distances to the test sample and normalize the distances between [0, 1]. Note that if the test point and the training one overlap in the space, the distance will be 0. If a training point is at the farthest location, its distance will be exactly 1 due to the normalization.
4. Split the laid samples into *l* groups using a basic clustering method (*k*-means),
5. Take all the samples in the closest cluster into the *k*-NN classifier and apply majority voting.

These steps are repeated for each test case. As the *k* parameter is generally set to small integer numbers in *k*-NN, it is set to 1 in the *k*-NC method as well.

Euclidean distance, as a most popular choice, is used to calculate the distance between two instances even though there are some other metrics that might be preferred. The general distance formula (5.1) between *x* and *y* is defined as follows:

$$d(x, y) = \left(\sum_{i=1}^{D} |a_i(x) - a_i(y)|^r \right)^{1/r} \tag{5.1}$$

where $a_i(x)$ denotes the value of the ith attribute of instance *x*. The arbitrary instance *x* in the dataset is described by the feature vector: $\langle a_1(x), a_2(x), a(x), \ldots a_D(x) \rangle$ where *D* is the number of dimensions (attributes). In this formula (5.1), if *r* is set to 1, it becomes *Manhattan* (*Cityblock*) distance; if it is set to 2, it is Euclidean; if it is set to more than 2, it turns into *Minkowsky* distance. In the limiting case of *r* reaching infinity, *Chebyshev* distance is obtained. As an alternate and versatile choice, *Mahalanobis* is another metric between a point and a distribution. Apart from the others, *Mahalanobis* puts emphasis to the distributions.

The function of *k*-NN learner (5.2) predicts the class of the query point x_q as:

$$\hat{f}(x_q) \leftarrow \underset{c \in C}{argmax} \sum_{i=1}^{k} \delta(c, f(x_i)) \tag{5.2}$$

where x_i … x_k denotes the *k* closest training samples to x_q, and $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise. The $f(x_i)$ function handles the closest *k* points as a target function. *C* is the finite set of class labels defined as $\{c_1, c_2, c_3, \ldots c_s\}$. The $f: R \rightarrow C$ function makes a calculation for each class and assigns the maximum argument to x_q as the class label.

k-NN, at the same time, can be used as a regression function as in the formula (5.3). Here, the f function calculates the mean value of the *k* nearest training samples in order to label the new query point x_q.

$$f(x_q) = \frac{1}{k} \sum_{i=1}^{k} f(x_i) \tag{5.3}$$

Dividing the *M* closest points into *l* clusters and contributing the nearest cluster to the calculation generate a dynamic structure. It is preferred to use the *l* abbreviation rather than *k* because it is the same in both *k*-means and *k*-NN. In each of the *l* clusters we can assume that there might be approximately M/l training points. The relationship between the k parameter and the M/l pair can be defined as in this equation:

$$k \cong \frac{M}{l} \tag{5.4}$$

The M/l combination provides a dynamic structure for each test case. The number of M/l training samples will be included in the prediction procedure. In Figure 5.18, there is a real illustration about M/l pairs from the experimental steps. M/l pair is $15/3$. There is a histogram of 15 nearest samples to a test point in the *abalone* dataset. The points in the histogram are divided into 3 clusters by means of a simple clustering method. In the 1NC technique, the closest 4 points in the nearest cluster will be used in the calculation of *k*-NN. In other words, the locally adapting *k* value for the current test point is set to 4.

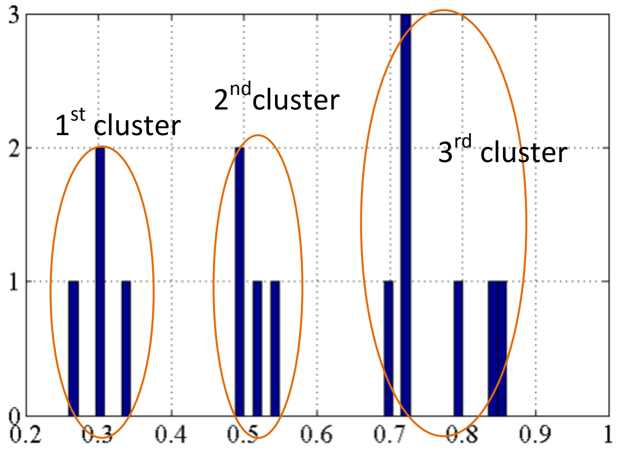

Figure 5.18 The histogram of nearest points to a test point example

5.9.1 Search Methods and Time Complexities

As an instance and memory based classifier, the *k*-NN method searches the nearest *k* points in the entire dataset for each test point during test process. This operation remarkably increases the execution time. In our experiment two types of search methods are preferred and implemented to decrease the computational time. If the number of dimension of a dataset is bigger than 10, Exhaustive search method is recommended; if it is smaller than 10, kD-Tree (*k Dimensional Tree*) searcher is recommended [112], since the construction time of kD-Tree increases as the number of dimension increases. Types of these search methods do not affect the classification performance, they only has an influence on the execution time. In Exhaustive search style, a point is found without using any algorithms and data structures. This method uses simple sequential search. The time complexity to find *k* closest points is as follows:

$$O(kDN) \tag{5.5}$$

where *D* is the number of dimension; *N* is the number of points [113]. kD-Tree is one of the BSP (*Binary Space Partitioning*) methods. kD-Tree is a multi-dimensional version of BST (*Binary Search Tree*) data structure. The time complexity of finding the *k* closest points is $O(kD\log N)$ due to the binary split in each dimension.

The complexity notation of *k*-means is as:

$$O(kNID) \tag{5.6}$$

where *k* and *I* are the numbers of clusters and iterations respectively [114].

The time complexity of the suggested method (1NC) is related to both the search and the clustering methods. Therefore, 1NC's complexity is slightly bigger than *k*-NN's because of the integration of searching, clustering and classification complexities. The overall complexity might be written as:

$$O\left(MD\log N + (\frac{M}{l})ID + (\frac{M}{l})\right) \tag{5.7}$$

In this notation, the first part $(MD\log N)$ is for searching the whole dataset to find the *M* closest points with kD-Tree searcher; the second part $\left((\frac{M}{l})ID\right)$ is for clustering; and the negligible third part $\left(\frac{M}{l}\right)$ is for majority voting in *k*-NN. Since the number of dimension *D* is a constant and small integer number, it becomes an insignificant parameter. Therefore, it is sometimes not written in the notations. Finally, the notation can be summarized as:

$$O\left(M\log N + (\frac{M}{l})I\right) \tag{5.8}$$

5.10 **EXPERIMENTAL RESULTS**

Experiments are performed on 36 UCI benchmark datasets [93] which represent a wide range of data characteristics, distributions and domains. In these real world datasets, missing values are replaced, nominal values are converted to binary values and finally all the values are normalized. 10 fold cross validation technique are used throughout the experiments on the MATLAB environment in order to investigate the performance of class probability estimation techniques. Despite the default value of the iteration number for *k*-means which is set to 100, it is decreased to 10 since the chosen nearest points in the experiments (remember *M*=15) are very few. It has been tested that whether the iteration number 10 is enough or not. Similar results have been reached by both the iteration numbers 100 and 10. The *k* parameter in *k*-NN is set to 5 to illustrate and test our technique. All accuracy rates are computed. Paired T-Test metric enables to compare the overall performances of the *k*-NN and 1NC classifiers statistically [115]. T-Test gives three types of results: *win* (if the first classifier is better), *loss* (worse) and *tie* (equal). All computational results are placed in Table 5.2.

In Table 5.2, the first three columns are the number of instances, attributes and the classes of each class respectively. These external features describe the density and sparseness of the feature space. In the next three columns, best *k*-NN, 1NN and 5NN results are presented. The next column has *One Nearest Cluster* classifier results where M/l combination is 15/3. This means the nearest 15 instances among the test point will be divided into 3 clusters. Accordingly there may be around 5 samples in each cluster. It means that the *k* value is set to the number of samples in the closest cluster. All samples in the nearest cluster will be computed in *k*-NN learning.

The accuracy comparison between 1NC $(M/l = 15/3)$and 5NN classifiers is done by T-Test and the results are seen in the "1NC-5NN comparison" column. In this comparison, there is up to %15 accuracy increment, 10 *wins*, 18 *ties*, and 8 *losses*. Additionally, 1NC and 1NN is compared in the last two columns: 12 *wins*, 20 *ties*, 4 *losses*. In the beginning, it has not been expected that 1NC would be as good as 1NN. These experimental results point out that 1NC outperforms 1NN significantly in these evaluations.

Table 5.3 presents min, max, mean number of points in the nearest cluster to the test point in the cross validation step. In each experiment, the minimum number of points is always equal to 1. The maximum number of points is always smaller than 15. Standard deviation of the points in the closest cluster is also placed in the table in order to give a notion about the dynamic parameter selection system.

Table 5. 2 Comparison of the methods over 36 UCI datasets

	External Features			Best *k*-NN results					1NC-5NN comparison		1NC-1NN comparison	
Dataset Name	# of inst.	# of Att.	# of cls.	*k*	Acc.	1NN Acc.	5NN Acc.	1NC *M*/*l*=15/3 Acc.	% incrs	T-Test	% incrs	T-Test
abalone	4153	19	10	55	0.2682	0.2023	0.2301	0.2216	-3.66	loss	9.55	win
anneal	890	4	62	1	0.9769	0.9769	0.9584	0.9699	1.20	tie	-0.71	tie
audiology	169	5	69	4	0.7053	0.6757	0.6852	0.6367	-7.08	loss	-5.78	loss
autos	202	5	71	1	0.6505	0.6505	0.5723	0.6188	8.13	win	-4.87	loss
balance-scl	625	3	4	100	0.8931	0.7894	0.8576	0.8579	0.04	tie	8.68	win
breast-cncr	286	2	38	9	0.7308	0.6643	0.7098	0.7049	-0.69	tie	6.11	win
breast-w	699	2	9	5	0.9671	0.9548	0.9671	0.9554	-1.21	tie	0.06	tie
col10	2019	10	7	1	0.7241	0.7241	0.7072	0.7256	2.61	win	0.10	tie
colic	368	2	60	74	0.8196	0.6957	0.7777	0.7245	-6.85	loss	4.13	win
credit-a	690	2	42	38	0.8452	0.7901	0.8304	0.8055	-3.00	loss	1.95	tie
credit-g	1000	2	59	14	0.7248	0.6824	0.7176	0.7022	-2.15	tie	2.90	win
d159	7182	2	32	1	0.9453	0.9453	0.9404	0.9485	0.87	tie	0.37	tie
diabetes	768	2	8	15	0.7466	0.6943	0.7286	0.7107	-2.47	tie	2.36	tie
glass	205	5	9	1	0.6713	0.6713	0.6410	0.6722	4.87	win	0.13	tie
heart-statlg	270	2	13	62	0.8356	0.7467	0.8052	0.7837	-2.67	tie	4.96	win
hepatitis	155	2	19	5	0.8361	0.7948	0.8361	0.7884	-5.71	loss	-0.81	tie
hypothyroid	3770	3	31	5	0.9329	0.9125	0.9329	0.9306	-0.26	tie	1.98	tie
ionosphere	351	2	33	1	0.8558	0.8558	0.8387	0.8638	2.99	win	0.46	tie
iris	150	3	4	10	0.9693	0.9467	0.9613	0.9533	-0.83	tie	0.70	tie
kr-vs-kp	3196	2	39	3	0.9008	0.8891	0.8923	0.9293	4.14	win	4.52	win
labor	57	2	26	1	0.8732	0.8732	0.8421	0.8611	2.25	win	-2.35	loss
letter	20000	26	16	1	0.9441	0.9441	0.9343	0.9404	0.65	tie	-0.36	tie
lymph	142	2	37	12	0.8338	0.7592	0.7915	0.7958	0.53	tie	4.82	win
mushroom	8124	2	112	1	1.0000	1.0000	0.9999	0.9999	0.00	tie	-0.01	tie
primary-tmr	302	11	23	18	0.4755	0.3874	0.4430	0.4159	-6.13	loss	7.36	win
ringnorm	7400	2	20	2	0.7915	0.7257	0.6623	0.7361	11.15	win	1.44	win
segment	2310	7	18	1	0.9580	0.9580	0.9443	0.9527	0.89	tie	-0.55	tie
sick	3772	2	31	5	0.9598	0.9569	0.9598	0.9563	-0.37	tie	-0.07	tie
sonar	208	2	60	1	0.8375	0.8375	0.7481	0.8212	9.77	win	-1.95	tie
soybean	675	18	83	1	0.8916	0.8916	0.8776	0.8830	0.61	tie	-0.97	tie
splice	3190	3	287	99	0.8413	0.7357	0.7285	0.7645	4.94	win	3.91	win
vehicle	846	4	18	3	0.6839	0.6723	0.6825	0.6589	-3.46	loss	-2.00	tie
vote	435	2	16	3	0.9297	0.9228	0.9297	0.9218	-0.84	tie	-0.10	tie
vowel	990	11	11	1	0.9473	0.9473	0.7806	0.8972	14.93	win	-5.29	loss
waveform	5000	3	40	75	0.8492	0.7278	0.7875	0.7519	-4.52	loss	3.31	win
zoo	84	4	16	1	0.9976	0.9976	0.9929	0.9857	-0.72	tie	-1.30	tie
MEAN					0.8281	0.7943	0.7970	0.8012				

Table 5. 3 min, max, mean number of points in the nearest cluster and standard deviation.

		M/*l* = 15/3			
No	Dataset Name	Min # of points in the cluster	Max # of points in the cluster	# of Mean points in the cluster	Standard Deviation
1	abalone	1	10	3.288	1.796
2	anneal	1	12	5.362	4.094
3	audiology	1	9	2.781	2.08
4	autos	1	10	7.347	4.697
5	balance-scl	1	11	6.282	4.822
6	breast-cncr	1	11	4.254	3.254
7	breast-w	1	10	3.547	1.985
8	col10	1	9	6.205	5.446
9	colic	1	9	2.825	1.895
10	credit-a	1	11	3.248	2.239
11	credit-g	1	10	2.753	1.741
12	d159	1	9	2.343	1.452
13	diabetes	1	8	2.982	1.678
14	glass	1	8	3.457	1.984
15	heart-statlg	1	11	3.399	2.066
16	hepatitis	1	11	3.457	2.463
17	hypothyroid	1	12	3.789	2.355
18	ionosphere	1	9	3.701	2.054
19	iris	1	9	3.881	2.053
20	kr-vs-kp	1	12	7.644	5.718
21	labor	1	13	3.001	2.024
22	letter	1	14	3.419	2.107
23	lymph	1	12	3.366	2.105
24	mushroom	1	9	3.987	1.654
25	primary-tmr	1	10	7.265	5.387
26	ringnorm	1	10	2.655	1.565
27	segment	1	10	3.407	1.961
28	sick	1	11	3.738	2.309
29	sonar	1	7	2.913	1.599
30	soybean	1	11	4.553	3.448
31	splice	1	12	5.382	4.569
32	vehicle	1	10	3.122	1.731
33	vote	1	14	8.894	6.276
34	vowel	1	8	2.633	1.498
35	waveform	1	9	2.722	1.577
36	zoo	1	8	2.875	1.661
	AVARAGE	1	10.3	4.068	2.704

It is certain that an instance based classifier requiring two hyper parameters is not better than *k*-NN including only one parameter. 1NC might be considered as a classifier requiring only one parameter by means of some experiments. These experiments handle some different M/l combinations such as 10/2, 15/3, 30/6, 50/10, 100/20, and 200/40. The results of the divisions are almost equal to 5. This means *k* is almost 5 in *k*-NN. Surprisingly, similar *win-tie-loss* results have been reached in different experiments as shown in Table 5. 4. These similar experimental outputs of the M/l combinations provide us three consequences (inferences):

1. M/l pairs should have smaller integer numbers since different *M* and *l* values give similar outputs as seen in Table 3.
2. Small *M* and *l* values decrease the overall computational time. The computational cost of clustering remains negligible owing to the small *M* and *l* values.
3. The M/l combination might be assumed as a unique hyper parameter of this method. If the *l* value is fixed, *M* value might be 4 or 5 times bigger than *l* (remember $k \cong M/l$). Consequently, a proximity between M/l and *k* parameters is established.

Table 5. 4 The effect of different M/l pairs

		M/l pairs					
		10/2	15/3	30/6	50/10	100/20	200/4
T-Test Results	number of *win*	7	10	8	7	8	8
	number of *tie*	22	18	21	24	22	22
	number of *loss*	7	8	7	5	6	6

In most cases, small *k* parameters in *k*-NN give higher accuracy rates. In the experiment, 100 sorts of *k*-NN classifier (*k* value is consecutively set from 1 to 100 in ascending order) are applied to 36 UCI datasets. As shown in Figure 5.19, the highest accuracy rates are computed in 14 datasets when k is equal to 1. It is particularly evident in the figure that small k values usually provide higher accuracy rates.

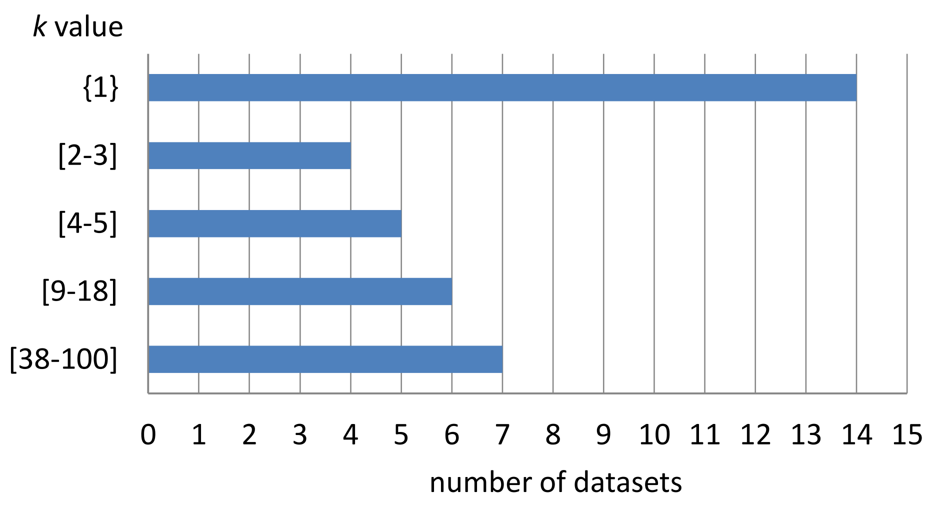

Figure 5.19 Best k value of k-NN on the 36 UCI datasets

5.11 **CONCLUSION**

In this research, a novel solution for dealing with the shortcomings of k-NN has been suggested. A novel way of assigning a k value best fitting to each test instance to acquire better performance is developed. The proposed research contributes a dynamic k parameter selector to the traditional k Nearest Neighbors algorithm.

A mechanism also is proposed with a strategy of combining lazy learning with an unsupervised learning method for two reasons. The first one is to eliminate the problems of the dependency on k without user's intervention and the latter is to augment the classification performance. The improved method using a simple clustering technique finds the appropriate k value for each test sample. The value of k during the classification phase varies dynamically. The well-suited dynamic k value which is assigned to each test data supplements more flexibility to the classification methodology. Experimental results carried out on 36 real world datasets show that the proposed hybrid k-NN Model is a competitive method for classification. With a single parameter (M/l), more successful prediction results are reached via experiments. Nevertheless, our supervised classification method has a little bit more time complexity since it contains an unsupervised method.

5.12 **AVAILABILITY**

The implemented MATLAB codes of the proposed model, the overall experimental results in Excel documents and related files can be publicly downloaded from the URL address for examinations and further studies:

https://www.ce.yildiz.edu.tr/personal/mfatih/file/4841/OneNC

CHAPTER 6

RESULTS AND DISCUSSION

In the essence of this thesis study, construction and performance analysis of locally adaptive base and ensemble learners have been proposed by using Meta and Ensemble Learning methods. The thesis stands on the three bases related with the characteristics and meta-features of the discretized sub regions in a dataset. The first one is the prediction and analysis of the decision tree classifier performance using Meta Learning methods by focusing on the local features of the sub regions in a dataset. The second one concerns a new approach in Mixture of Experts using hard clustering techniques in order to establish successful experts for the discrete sub partitions. Finally, the third one is about a locally adaptive parameter selection mechanism for memory based classifiers using clustering algorithms.

Firstly, prediction of decision tree classifier performance is proposed by means of Meta Learning methods. We have tried to guess the performance and behavior of DT classifier on a dataset. Also, we have tried to find out why and how DTs outperform or fail on a dataset and what kind of features contribute to the performance of a DT. For this purpose, a collection of two-class datasets is used for our empirical study. 14 meta-features have been derived from the collection. Only two features (the LOO Error rate of 1NN classifier and the Error rate of linear classifier) are highly influential on prediction activity. In other words, these two meta-features mainly affect the performance of DTs. After building the formula of linear regression model, we have tried to understand how and why the meta-features affect the accuracy of DT. We have taken sufficient and successful results such as a meaningful and feasible regression formula including meta-features. Thus, it becomes easy to predict the performance of a DT classifier. Briefly, this study has yielded two main ideas. First, the higher the gap in the boundaries of classes is, the higher performance we get from DTs' predictions. Second, building a linear classifier that bisects the class distributions with less error rate remarkably increases the performance of DT classifier.

Secondly, a new approach in Mixture of Experts is presented for accurate prediction and classification. In contrast to the traditional Mixture of Experts method, in this theoretical and practical study, a dataset is divided into partitions by a hard clustering method and the class prediction method is performed by four different types of proposed gate functions: cooperating, competitive, commensurative, and Borda count. In the experiments, better performances have been obtained with the proposed cooperating gate function due to its mechanism that gives different weights to the experts in the network. Typically, MoE model is particularly advantageous when different types of experts are trained on different parts of the same space. Training a set of chosen experts and assigning the best ones to each region might be another expensive choice in the training phase. But in this study the same type of experts are chosen rather than heterogeneous ones throughout the experiments. It is also showed that the total classification performance might be increased with this suggested MoE approach using a cooperative gate function and homogeneous experts on hard clustered regions. In addition, this cognitive study enables the proposed MoE model to be applicable to any type of dataset unlike the other models like GMM.

Thirdly, locally adaptive parameter selection mechanism for memory based classifiers is proposed with a strategy of combining lazy learning with an unsupervised learning method for two reasons. The first one is to eliminate the problems of the dependency on *k* without the user's intervention and the second one is to increase the classification performance. The improved method using a simple clustering technique finds the appropriate *k* value for each test sample. The value of *k* during the classification phase varies dynamically. The well-suited dynamic *k* value which is assigned to each test data adds more flexibility to the classification methodology. More successful prediction results are reached in the experiments carried out on some real world datasets. The reasons of success have also been understood and presented.

In conclusion, the characteristics and meta-features of the discrete sub regions in a training dataset have great importance in classification phase. It is both theoretically and empirically proved that some individual and different local specifications affect the overall classification performance.

APPENDIX-A

META LEARNING DATASET

In the Table 7.1 below, there are external rough features describing a dataset. These features are the number of instances, the number of attributes, the number of c1 class labels, and the number of c2 class labels respectively.

In the Table 7.2, for each dataset there are 14 meta-features described in Chapter 3.

In the Table 7.3, the actual and predicted DT accuracy values are presented. The predicted accuracy levels are computed by the linear regression model. The error rates show the differences between actual and predicted accuracies.

Table 7. 1 The external features of the datasets

DATA SET	Number of Instances	Number of Attributes	Number of c1 class labels	Number of c2 class labels
hillValley	606	100	301	305
bank	300	11	162	138
liver-disorders	345	6	145	200
bupa	345	6	145	200
cmc.2c2	1472	9	962	510
liv	345	6	145	200
cmc.2c2	1472	9	962	510
bpa	345	6	145	200
hab	306	3	225	81
cmc.2c0	1472	9	843	629
breast-cancer	286	38	201	85
haberman	306	3	225	81
credit-g	1000	59	700	300
yea.2c0	1484	8	1021	463
cylinder-bands	540	896	312	228
sonar	208	60	97	111
pim	768	8	500	268
glass.2c1	204	9	129	75
diabetes	768	8	500	268
lung-cancer	32	56	9	23
cmc.2c1	1472	9	1139	333
cmc.2c1	1472	9	1139	333
transfusion	748	4	570	178
vehicle.2c1	845	18	628	217
h-s	270	13	150	120
abalone.2c6	4152	10	3463	689
vehicle.2c0	845	18	633	212
veh.2c0	846	18	634	212
heart-statlog	270	13	150	120
abalone.2c7	4152	10	3518	634
gls.2c0	214	9	144	70
glass.2c0	204	9	134	70
colic	368	60	232	136
hepatitis	155	19	32	123
primary-tumor.2c0	301	23	217	84
abalone.2c5	4152	10	3584	568

Table 7. 1 The external features of the datasets (continuous)

column3C.2c0	309	6	249	60
column3C.2c2	309	6	210	99
lymph	142	37	81	61
abalone.2c8	4152	10	3665	487
waveform.2c0	4999	40	3308	1691
wav40.2c0	5000	40	1692	3308
wav21.2c0	5000	21	1657	3343
mag	19020	10	12332	6688
autos.2c1	201	71	134	67
balance-scale.2c0	624	4	336	288
autos.2c2	201	71	147	54
bankruptcy	50	5	25	25
waveform.2c2	4999	40	3344	1655
bal.2c0	625	4	337	288
waveform.2c1	4999	40	3346	1653
credit-a	690	42	307	383
abalone.2c4	4152	10	3761	391
glass.2c2	204	9	187	17
labor	57	26	20	37
ionosphere	351	33	126	225
col10.2c4	2018	7	1774	244
ecoli.2c1	326	6	249	77
audiology.2c3	168	69	120	48
ringnorm	7400	20	3664	3736
col10.2c5	2018	7	1704	314
spambase	4601	57	1813	2788
tic-tac-toe	958	27	332	626
ecoli.2c3	326	6	291	35
audiology.2c4	168	69	146	22
balance-scale.2c1	624	4	576	48
spa	4601	57	2788	1813
monk	122	6	60	62
thy.2c0	215	5	65	150
ecoli.2c2	326	6	275	51
vehicle.2c3	845	18	647	198
wineCultivars.2c1	152	13	91	61
wdbc	569	30	212	357
ecoli.2c0	326	6	183	143
wne.2c0	178	13	119	59

Table 7. 1 The external features of the datasets (continuous)

wineCultivars.2c2	152	13	108	44
wineCultivars.2c0	152	13	105	47
vote	435	16	267	168
win.2c0	178	13	119	59
iris.2c1	149	4	99	50
splice.2c2	3189	287	1535	1654
authors.2c0	841	70	317	524
vehicle.2c2	845	18	627	218
iris.2c2	149	4	100	49
tao	1888	2	944	944
ozone	2536	72	2463	73
zoo.2c2	83	16	70	13
column3C.2c1	309	6	159	150
audiology.2c0	168	69	112	56
pageblocks.2c0	5472	10	559	4913
pbc.2c0	5473	10	560	4913
ecoli.2c4	326	6	306	20
solar-flare_1	323	32	316	7
d159	7182	32	3719	3463
sick	3772	31	3541	231
pageblocks.2c4	5472	10	5357	115
pageblocks.2c1	5472	10	5143	329
anneal.2c1	889	62	205	684
kr-vs-kp	3196	39	1669	1527
opt.2c0	5620	64	554	5066
statlog-sgm.2c0	2310	19	1980	330
seg.2c0	2310	19	1980	330
col10.2c6	2018	7	1724	294
zoo.2c0	83	16	42	41
soybean.2c3	674	83	586	88
pageblocks.2c3	5472	10	5385	87
pen.2c0	10992	16	1143	9849
pageblocks.2c2	5472	10	5444	28
hypothyroid.2c0	3769	31	289	3480
mushroom	8124	112	4208	3916
badges	294	10	84	210
badges2	294	10	84	210
col10.2c0	2018	7	1415	603
iris.2c0	149	4	99	50
zoo.2c3	83	16	73	10

Table 7. 2 The normalized meta-features of the datasets

DATA SET	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
hillValley	0.000	0.000	0.000	0.013	0.997	0.502	1.000	1.000	1.000	0.982	1.000	1.000	1.000	0.003
bank	0.004	0.000	0.883	0.028	0.047	0.420	0.746	0.760	0.968	0.841	0.940	0.823	0.950	0.014
liver-disorders	0.003	0.000	0.073	0.033	0.107	0.426	0.847	1.000	0.930	0.942	0.895	0.662	1.000	0.030
bupa	0.003	0.000	0.073	0.033	0.107	0.426	0.847	1.000	0.930	0.942	0.895	0.681	1.000	0.030
cmc.2c2	0.005	0.000	0.812	0.001	0.001	0.351	0.698	1.000	0.958	0.871	0.976	0.893	0.945	0.086
liv	0.003	0.000	0.073	0.033	0.107	0.426	0.847	1.000	0.930	0.942	0.895	0.712	1.000	0.030
cmc.2c2	0.005	0.000	0.812	0.001	0.001	0.351	0.698	1.000	0.958	0.871	0.976	0.867	0.945	0.086
bpa	0.003	0.000	0.073	0.033	0.107	0.426	0.847	1.000	0.930	0.942	0.895	0.749	1.000	0.030
hab	0.009	0.000	0.718	0.030	0.033	0.269	0.534	1.000	0.873	0.763	0.844	0.891	0.885	0.053
cmc.2c0	0.006	0.000	0.750	0.002	0.002	0.400	0.718	0.802	0.946	0.910	0.990	0.865	0.920	0.086
breast-cancer	0.014	0.001	0.000	0.014	0.021	0.317	0.474	0.894	0.773	0.879	0.840	0.467	1.000	0.004
haberman	0.078	0.000	0.718	0.030	0.033	0.269	0.534	1.000	0.822	0.637	0.837	0.806	0.766	0.053
credit-g	0.016	0.001	0.662	0.014	0.024	0.378	0.468	0.668	0.724	0.914	0.677	0.264	1.000	0.009
yea.2c0	0.011	0.000	0.056	0.130	0.177	0.316	0.629	1.000	0.729	0.714	0.708	0.766	1.000	0.097
cylinder-bands	0.006	0.021	0.000	0.060	0.996	0.275	0.389	0.252	0.690	0.837	0.469	0.009	1.000	0.000
sonar	0.024	0.004	0.000	0.054	1.000	0.327	0.389	0.228	0.469	0.749	0.301	0.238	1.000	0.002
pim	0.029	0.001	0.252	0.007	0.022	0.349	0.706	0.998	0.710	0.860	0.703	0.609	0.998	0.050
glass.2c1	0.007	0.000	0.009	0.105	0.328	0.373	0.742	1.000	0.604	0.499	0.493	0.701	0.992	0.012
diabetes	0.029	0.001	0.252	0.007	0.022	0.349	0.708	1.000	0.710	0.860	0.706	0.648	0.998	0.050
lung-cancer	0.031	0.013	0.000	0.287	0.906	0.217	0.442	0.812	0.760	0.973	0.897	0.000	1.000	0.000
cmc.2c1	0.012	0.000	0.727	0.068	0.069	0.230	0.456	1.000	0.731	0.686	0.749	0.814	0.935	0.086
cmc.2c1	0.012	0.000	0.727	0.068	0.069	0.230	0.456	1.000	0.731	0.686	0.749	0.823	0.935	0.086
transfusion	0.015	0.000	0.271	0.008	0.012	0.242	0.480	1.000	0.698	0.628	0.792	0.838	0.975	0.098
vehicle.2c1	0.008	0.002	0.000	0.061	0.218	0.264	0.518	1.000	0.602	0.746	0.615	0.749	0.998	0.024
h-s	0.039	0.002	0.196	0.015	0.093	0.270	0.315	0.238	0.594	0.672	0.584	0.341	1.000	0.011
abalone.2c6	0.002	0.000	0.017	0.072	0.091	0.168	0.335	1.000	0.606	0.695	0.615	0.987	0.990	0.218
vehicle.2c0	0.009	0.001	0.001	0.038	0.222	0.255	0.506	1.000	0.589	0.716	0.596	0.793	0.998	0.024
veh.2c0	0.009	0.001	0.001	0.038	0.223	0.255	0.506	1.000	0.591	0.716	0.593	0.775	0.998	0.024
heart-statlog	0.040	0.003	0.196	0.015	0.093	0.270	0.315	0.174	0.596	0.673	0.586	0.284	1.000	0.011
abalone.2c7	0.008	0.000	0.044	0.068	0.078	0.155	0.308	1.000	0.555	0.639	0.565	0.980	0.952	0.218
gls.2c0	0.033	0.000	0.000	0.296	0.486	0.340	0.659	1.000	0.521	0.402	0.435	0.382	0.985	0.012
glass.2c0	0.030	0.001	0.000	0.265	0.422	0.356	0.692	1.000	0.525	0.413	0.457	0.424	0.992	0.012
colic	0.068	0.003	0.187	0.039	0.098	0.295	0.274	0.280	0.737	0.873	0.770	0.122	1.000	0.003

Table 7. 2 The normalized meta-features of the datasets (continuous)

hepatitis	0.039	0.003	0.000	0.239	0.565	0.229	0.405	1.000	0.463	0.635	0.450	0.255	1.000	0.004
primary-tumor.2c0	0.031	0.003	1.000	0.000	0.000	0.271	0.335	0.564	0.598	0.645	0.517	0.373	1.000	0.007
abalone.2c5	0.009	0.000	0.013	0.086	0.117	0.139	0.276	1.000	0.484	0.530	0.500	0.945	0.978	0.218
column3C.2c0	0.059	0.002	0.001	0.492	0.686	0.205	0.391	1.000	0.393	0.457	0.340	0.402	0.978	0.027
column3C.2c2	0.038	0.001	0.005	0.366	0.450	0.329	0.645	1.000	0.513	0.632	0.457	0.384	0.995	0.027
lymph	0.051	0.006	0.000	0.051	0.113	0.242	0.200	0.134	0.564	0.830	0.423	0.070	1.000	0.002
abalone.2c8	0.017	0.000	0.026	0.063	0.077	0.119	0.236	1.000	0.460	0.605	0.471	0.969	0.942	0.218
waveform.2c0	0.060	0.002	0.015	0.152	0.211	0.507	0.288	0.160	0.442	0.927	0.469	0.140	1.000	0.065
wav40.2c0	0.060	0.000	0.015	0.152	0.211	0.507	0.288	0.150	0.442	0.927	0.469	0.140	1.000	0.065
wav21.2c0	0.060	0.000	0.036	0.126	0.183	0.517	0.284	0.174	0.385	0.809	0.407	0.238	1.000	0.125
mag	0.029	0.005	0.081	0.006	0.018	0.363	0.429	0.472	0.474	0.647	0.450	0.513	1.000	1.000
autos.2c1	0.040	0.005	0.000	0.193	1.000	0.283	0.452	0.622	0.460	0.446	0.273	0.227	0.983	0.001
balance-scale.2c0	0.020	0.004	1.000	0.000	0.000	0.288	0.097	0.142	0.388	0.615	0.304	0.225	0.837	0.082
autos.2c2	0.030	0.004	0.000	0.213	0.925	0.282	0.542	0.990	0.476	0.451	0.356	0.325	0.983	0.001
bankruptcy	0.057	0.001	0.001	0.633	0.960	0.467	0.766	0.820	0.388	0.627	0.335	0.153	0.867	0.005
waveform.2c2	0.072	0.002	0.001	0.245	0.383	0.675	0.216	0.078	0.374	0.868	0.397	0.087	1.000	0.065
bal.2c0	0.020	0.000	1.000	0.000	0.000	0.289	0.097	0.188	0.297	0.617	0.330	0.183	0.837	0.082
waveform.2c1	0.068	0.002	0.001	0.239	0.408	0.687	0.222	0.084	0.385	0.878	0.404	0.100	1.000	0.065
credit-a	0.115	0.004	0.000	0.034	0.068	0.147	0.292	0.284	0.552	0.530	0.471	0.303	0.998	0.008
abalone.2c4	0.045	0.001	0.002	0.122	0.203	0.096	0.190	1.000	0.319	0.346	0.342	0.830	0.988	0.218
glass.2c2	0.018	0.001	0.000	0.465	0.824	0.085	0.167	1.000	0.278	0.343	0.282	0.686	0.992	0.012
labor	0.069	0.010	0.000	0.218	0.964	0.175	0.252	0.322	0.289	0.539	0.256	0.079	1.000	0.001
ionosphere	0.031	0.004	0.000	0.195	0.994	0.231	0.236	0.292	0.374	0.626	0.313	0.384	0.915	0.005
col10.2c4	0.008	0.000	0.000	0.344	0.578	0.123	0.244	1.000	0.297	0.124	0.306	0.762	0.839	0.151
ecoli.2c1	0.143	0.004	0.150	0.404	0.521	0.248	0.476	1.000	0.293	0.386	0.301	0.314	0.882	0.028
audiology.2c3	0.102	0.019	0.000	0.486	0.970	0.238	0.192	0.410	0.559	0.641	0.414	0.247	1.000	0.001
ringnorm	0.003	0.001	0.000	0.061	0.418	0.364	0.452	0.378	0.719	0.874	0.591	0.624	1.000	0.194
col10.2c5	0.011	0.001	0.000	0.379	0.490	0.158	0.315	1.000	0.278	0.143	0.268	0.692	0.839	0.151
spambase	0.011	0.001	0.000	0.105	0.430	0.344	0.673	0.900	0.374	0.426	0.299	0.402	0.967	0.042
tic-tac-toe	0.014	0.034	1.000	0.000	0.000	0.227	0.236	0.438	0.838	1.000	0.780	0.400	1.000	0.018
ecoli.2c3	0.081	0.003	0.049	0.563	0.767	0.120	0.216	1.000	0.237	0.240	0.256	0.395	0.914	0.028
audiology.2c4	0.105	0.022	0.000	0.595	1.000	0.250	0.155	0.560	0.346	0.554	0.227	0.092	1.000	0.001
balance-scale.2c1	0.000	0.000	1.000	0.000	0.000	0.078	0.155	1.000	0.366	0.648	0.376	0.910	0.814	0.082
spa	0.018	0.006	0.000	0.093	0.383	0.391	0.794	1.000	0.250	0.340	0.196	0.238	0.977	0.042
monk	0.031	0.002	1.000	0.000	0.000	0.327	0.351	0.248	0.656	0.752	0.514	0.334	0.932	0.010

Table 7. 2 The normalized meta-features of the datasets (continuous)

thy.2c0	0.013	0.000	0.001	0.190	0.414	0.298	0.609	1.000	0.164	0.266	0.067	0.264	0.729	0.022
ecoli.2c2	0.092	0.004	0.000	0.226	0.577	0.176	0.315	1.000	0.138	0.387	0.139	0.190	0.949	0.028
vehicle.2c3	0.057	0.006	0.000	0.467	0.747	0.267	0.472	1.000	0.195	0.386	0.141	0.356	1.000	0.024
wineCultivars.2c1	0.112	0.010	0.001	0.598	1.000	0.279	0.452	0.578	0.202	0.554	0.093	0.050	1.000	0.006
wdbc	0.174	0.031	0.000	0.528	0.998	0.273	0.099	0.042	0.115	0.544	0.112	0.026	0.997	0.010
ecoli.2c0	0.143	0.008	0.000	0.522	0.770	0.215	0.093	0.042	0.134	0.360	0.132	0.096	0.929	0.028
wne.2c0	0.220	0.013	0.000	0.780	1.000	0.188	0.113	0.056	0.108	0.467	0.067	0.024	0.990	0.007
wineCultivars.2c2	0.263	0.020	0.000	0.726	0.987	0.173	0.052	0.066	0.074	0.404	0.031	0.007	0.967	0.006
wineCultivars.2c0	0.241	0.017	0.000	0.859	1.000	0.209	0.185	0.368	0.127	0.466	0.062	0.028	1.000	0.006
vote	0.548	0.014	1.000	0.000	0.000	0.161	0.089	0.058	0.211	0.360	0.182	0.039	1.000	0.014
win.2c0	0.220	0.013	0.000	0.780	1.000	0.188	0.113	0.078	0.108	0.467	0.067	0.044	0.990	0.007
iris.2c1	0.035	0.001	0.035	0.576	0.893	0.342	0.677	1.000	0.163	0.155	0.112	0.367	0.822	0.019
splice.2c2	0.077	0.007	0.000	0.001	0.004	0.724	0.081	0.044	0.723	0.849	0.593	0.001	1.000	0.006
authors.2c0	0.096	0.011	0.000	0.230	0.998	0.209	0.004	0.000	0.018	0.784	0.010	0.000	1.000	0.006
vehicle.2c2	0.019	0.007	0.003	0.232	0.696	0.267	0.520	1.000	0.148	0.519	0.093	0.362	0.997	0.024
iris.2c2	0.203	0.004	0.007	0.773	0.919	0.314	0.583	0.920	0.140	0.104	0.112	0.037	0.699	0.019
tao	0.071	0.000	0.479	0.367	0.362	0.299	0.329	0.230	0.124	0.094	0.103	0.356	0.000	0.496
ozone	0.049	0.003	0.000	0.322	0.912	0.030	0.058	1.000	0.111	0.604	0.117	0.585	1.000	0.018
zoo.2c2	0.247	0.202	0.000	0.701	0.687	0.039	0.000	0.000	0.096	0.073	0.029	0.013	1.000	0.002
column3C.2c1	0.077	0.003	0.005	0.595	0.773	0.303	0.359	0.236	0.330	0.552	0.301	0.155	1.000	0.027
audiology.2c0	0.188	0.026	0.000	0.535	0.988	0.186	0.036	0.030	0.473	0.598	0.342	0.046	1.000	0.001
pageblocks.2c0	0.026	0.005	0.000	0.016	0.026	0.161	0.192	0.968	0.109	0.124	0.089	0.452	0.980	0.287
pbc.2c0	0.026	0.008	0.000	0.016	0.026	0.162	0.192	0.972	0.109	0.124	0.089	0.472	0.980	0.288
ecoli.2c4	0.167	0.009	0.004	0.598	0.991	0.079	0.123	1.000	0.054	0.184	0.050	0.087	0.822	0.028
solar-flare_1	0.037	0.006	0.000	0.250	0.700	0.030	0.044	1.000	0.099	0.205	0.103	0.537	1.000	0.005
d159	0.042	0.466	0.000	0.033	0.127	0.045	0.000	0.036	0.147	0.574	0.062	0.153	0.988	0.118
sick	0.103	0.003	0.000	0.141	0.307	0.074	0.123	1.000	0.108	0.107	0.089	0.371	0.960	0.064
pageblocks.2c4	0.085	0.006	0.000	0.287	0.319	0.024	0.042	1.000	0.044	0.059	0.043	0.557	0.711	0.287
pageblocks.2c1	0.058	0.009	0.000	0.018	0.044	0.106	0.117	0.986	0.054	0.100	0.036	0.269	0.980	0.287
anneal.2c1	0.074	0.019	0.000	0.077	0.857	0.139	0.165	0.232	0.061	0.103	0.014	0.234	0.982	0.007
kr-vs-kp	0.027	0.004	0.000	0.187	0.433	0.390	0.125	0.160	0.443	0.709	0.371	0.297	1.000	0.043
opt.2c0	0.242	0.000	0.000	0.481	0.865	0.847	0.004	0.006	0.000	0.376	0.000	0.009	0.985	0.046
statlog-sgm.2c0	0.092	0.006	0.000	0.749	1.000	0.246	0.288	1.000	0.015	0.046	0.007	0.083	0.889	0.064
seg.2c0	0.092	0.010	0.000	0.749	1.000	0.246	0.288	1.000	0.015	0.046	0.007	0.083	0.889	0.064
col10.2c6	0.079	0.010	0.001	0.752	0.983	0.171	0.103	0.290	0.043	0.017	0.022	0.039	0.844	0.151

Table 7. 2 The normalized meta-features of the datasets (continuous)

zoo.2c0	1.000	0.149	0.000	0.480	0.470	0.033	0.000	0.000	0.077	0.071	0.000	0.000	1.000	0.002
soybean.2c3	0.286	0.095	0.000	0.752	0.988	0.240	0.000	0.000	0.013	0.370	0.007	0.002	1.000	0.004
pageblocks.2c3	0.143	0.012	0.000	0.609	0.752	0.026	0.032	1.000	0.017	0.012	0.017	0.415	0.944	0.287
pen.2c0	0.178	0.000	0.755	0.111	0.120	1.000	0.028	0.098	0.000	0.101	0.001	0.044	0.832	0.361
pageblocks.2c2	0.105	0.063	0.000	0.994	1.000	0.016	0.010	1.000	0.005	0.000	0.005	0.321	0.636	0.287
hypothyroid.2c0	0.035	0.003	0.000	0.927	0.926	0.089	0.155	1.000	0.200	0.210	0.191	0.605	0.973	0.064
mushroom	0.169	1.000	0.000	0.271	0.997	0.180	0.002	0.004	0.004	0.364	0.000	0.004	1.000	0.038
badges	0.017	0.001	0.071	1.000	0.980	0.000	0.000	0.000	0.022	0.004	0.000	0.000	0.502	0.015
badges2	0.003	0.001	0.071	1.000	0.980	0.000	0.000	0.000	0.022	0.004	0.000	0.000	0.503	0.015
col10.2c0	0.431	0.008	0.000	0.492	0.482	0.315	0.371	0.538	0.007	0.043	0.001	0.028	0.492	0.151
iris.2c0	0.854	0.024	0.005	0.582	0.570	0.158	0.000	0.000	0.020	0.026	0.000	0.000	0.096	0.019
zoo.2c3	0.846	0.170	0.000	0.811	0.904	0.097	0.000	0.000	0.038	0.064	0.000	0.026	1.000	0.002

Table 7. 3 The comparison between actual and predicted DT accuracies

DATA SET	Actual DT Accuracy	Predicted DT Accuracy	Error
hillValley	0.564	0.546	-0.02
bank	0.599	0.639	0.04
liver-disorders	0.605	0.642	0.04
bupa	0.619	0.639	0.02
cmc.2c2	0.619	0.628	0.01
liv	0.623	0.64	0.02
cmc.2c2	0.625	0.624	0.00
bpa	0.637	0.637	0.00
hab	0.650	0.694	0.04
cmc.2c0	0.650	0.622	-0.03
breast-cancer	0.660	0.702	0.04
haberman	0.667	0.696	0.03
credit-g	0.684	0.749	0.07
yea.2c0	0.692	0.72	0.03
cylinder-bands	0.693	0.803	0.11
sonar	0.695	0.7	0.00
pim	0.695	0.863	0.17
glass.2c1	0.706	0.771	0.07
diabetes	0.711	0.709	0.00
lung-cancer	0.713	0.653	-0.06
cmc.2c1	0.717	0.728	0.01
cmc.2c1	0.728	0.711	-0.02
transfusion	0.728	0.721	-0.01
vehicle.2c1	0.738	0.755	0.02
h-s	0.739	0.79	0.05
abalone.2c6	0.748	0.781	0.03
vehicle.2c0	0.751	0.755	0.00
veh.2c0	0.756	0.756	0.00
heart-statlog	0.761	0.791	0.03
abalone.2c7	0.765	0.799	0.03
gls.2c0	0.769	0.795	0.03
glass.2c0	0.778	0.789	0.01
colic	0.791	0.734	-0.06
hepatitis	0.794	0.828	0.04
primary-tumor.2c0	0.799	0.813	0.01
abalone.2c5	0.801	0.813	0.01
column3C.2c0	0.803	0.786	-0.02
column3C.2c2	0.803	0.858	0.06

Table 7. 3 The comparison between actual and predicted DT accuracies (continuous)

lymph	0.809	0.864	0.06
abalone.2c8	0.810	0.829	0.02
waveform.2c0	0.810	0.835	0.03
wav40.2c0	0.812	0.83	0.02
wav21.2c0	0.813	0.849	0.04
mag	0.817	0.813	0.00
autos.2c1	0.819	0.853	0.03
balance-scale.2c0	0.822	0.902	0.08
autos.2c2	0.828	0.858	0.03
bankruptcy	0.828	0.809	-0.02
waveform.2c2	0.828	0.819	-0.01
bal.2c0	0.831	0.893	0.06
waveform.2c1	0.831	0.865	0.03
credit-a	0.832	0.824	-0.01
abalone.2c4	0.856	0.878	0.02
glass.2c2	0.858	0.892	0.03
labor	0.860	0.863	0.00
ionosphere	0.871	0.834	-0.04
col10.2c4	0.874	0.878	0.00
ecoli.2c1	0.883	0.867	-0.02
audiology.2c3	0.883	0.854	-0.03
ringnorm	0.887	0.887	0.00
col10.2c5	0.887	0.735	-0.15
spambase	0.890	0.821	-0.07
tic-tac-toe	0.892	0.727	-0.16
ecoli.2c3	0.895	0.901	0.01
audiology.2c4	0.895	0.921	0.03
balance-scale.2c1	0.897	0.859	-0.04
spa	0.904	0.839	-0.07
monk	0.907	0.806	-0.10
thy.2c0	0.912	0.908	0.00
ecoli.2c2	0.914	0.914	0.00
vehicle.2c3	0.915	0.9	-0.02
wineCultivars.2c1	0.917	0.923	0.01
wdbc	0.925	0.95	0.03
ecoli.2c0	0.928	0.951	0.02
wne.2c0	0.934	0.974	0.04
wineCultivars.2c2	0.937	0.98	0.04
wineCultivars.2c0	0.938	0.967	0.03

Table 7. 3 The comparison between actual and predicted DT accuracies (continuous)

vote	0.941	0.934	-0.01
win.2c0	0.942	0.974	0.03
iris.2c1	0.942	0.889	-0.05
splice.2c2	0.944	0.97	0.03
authors.2c0	0.944	0.797	-0.15
vehicle.2c2	0.945	0.897	-0.05
iris.2c2	0.949	0.907	-0.04
tao	0.949	0.93	-0.02
ozone	0.952	0.938	-0.01
zoo.2c2	0.954	1.011	0.06
column3C.2c1	0.956	0.858	-0.10
audiology.2c0	0.961	0.875	-0.09
pageblocks.2c0	0.967	0.954	-0.01
pbc.2c0	0.967	0.952	-0.02
ecoli.2c4	0.969	0.967	0.00
solar-flare_1	0.971	0.95	-0.02
d159	0.971	0.982	0.01
sick	0.983	0.985	0.00
pageblocks.2c4	0.983	0.949	-0.03
pageblocks.2c1	0.985	0.993	0.01
anneal.2c1	0.986	0.947	-0.04
kr-vs-kp	0.988	0.859	-0.13
opt.2c0	0.989	0.991	0.00
statlog-sgm.2c0	0.989	0.979	-0.01
seg.2c0	0.990	0.965	-0.03
col10.2c6	0.990	1.01	0.02
zoo.2c0	0.990	0.987	0.00
soybean.2c3	0.993	0.987	-0.01
pageblocks.2c3	0.995	0.994	0.00
pen.2c0	0.996	1.06	0.06
pageblocks.2c2	0.996	1.02	0.02
hypothyroid.2c0	0.997	0.938	-0.06
mushroom	1.000	1.008	0.01
badges	1.000	0.996	0.00
badges2	1.000	0.962	-0.04
col10.2c0	1.000	1.014	0.01
iris.2c0	1.000	1.004	0.00
zoo.2c3	1.000	1.001	0.00

REFERENCES

[1] Song Q., Wang G. and Wang C., (2012). "Automatic recommendation of classification algorithms based on data set characteristics", Pattern Recognition, 45(7):2672–2689.

[2] Krijthe J.H., Ho T.K. and Loog M., (2012). "Improving Cross-validation Based Classifier Selection using Meta-Learning", in Pattern Recognition (ICPR), 21st International Conference on, 2873 – 2876, 11-15 Nov. 2012, Tsukuba.

[3] Luengo, J. and Herrera F., (2010). "An extraction method for the characterization of the Fuzzy Rule Based Classification Systems' behavior using data complexity measures: A case of study with FH-GBML", in Fuzzy Systems (FUZZ), IEEE International Conference on, 18-23 July 2010, Barcelona.

[4] Ricardo, V. and Youssef, D., (2002). "A Perspective View and Survey of Meta-Learning", Artificial Intelligence Review, Kluwer Academic Publishers Norwell, 18(2):77 - 95.

[5] Schmidhuber, T. and Schau J., (2010). Meta Learning.: Scholarpedia 5(6):4650.

[6] Vilalta, R. and Drissi, Y., (2006). "A perspective view and survey of meta-learning", Artificial Intelligence Review, 18(2):77—95.

[7] Smith-Miles, K.A., (2009). "Cross-Disciplinary Perspectives on MetaLearning for Algorithm Selection", ACM Computing Surveys (CSUR), 41(1):Article 6.

[8] Ali, S. and Smith, K.A., (2006). "On Learning Algorithm Selection for Classification", Applied Soft Computing, Elsevier Science, 6(5):119-138.

[9] Soares, C. and Brazdil, P.B. and Kuba, P., (2004). "A Meta-Learning Method

to Select the Kerne lWidth in Support Vector", Machine Learning, 54(1):195–209.

[10] Brazdil, P.B., Soares, C. and Costa, J.P., (2003). "Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results", Machine Learning, 50(3):251-277.

[11] Pan, S.J., (2009). "A Survey on Transfer Learning", Knowledge and Data Engineering, IEEE Transactions, 22(10) 1345 - 1359.

[12] Amasyali, M.F. ,and Ersoy, O., (2009). "A Study of Meta Learning for Regression", Technical Report, Purdue University, USA.

[13] Jacobs, R.A., Jordan, M.I., Nowlan, J. and Hinton, G. E., (1991). "Adaptive mixtures of local experts", Neural Computer, 3(1):79–87.

[14] Kabirc,E., Estekya, and H., Ebrahimpoura, (2008). "View-independent face recognition with Mixture of Experts", Neural Networks: Algorithms and Applications, 4th International Symposium on Neural Networks, 71(4-6):1103–1107.

[15] Xu, L. and Amar, S., (2009). Combining Classifiers and Learning Mixture-of-Experts, Encyclopedia of Artificial Intelligence (318-326).: Hershey, PA: Information Science Reference.

[16] Jordan, I. M. and Jacobs, A.R, (1994). "Hierarchical mixtures of experts and the EM algorithm", Neural Comput., 6(2):181–214.

[17] Yuksel, S.E., Wilson, J.N. and Gader P.D., (2012). "Twenty Years of Mixture of Experts", IEEE Transactions on Neural Networks and Learning Systems, 23(8):1177-1193.

[18] Masoudnia, S. and Ebrahimpour, R., (2014). "Mixture of experts: a literature survey", Springer Artificial Intelligence Review, 42(2):271-293.

[19] Ghosh, A.K., Chaudhuri, P. and Murthy, C. A., (2006). "Multi-scale Classification Using Nearest Neighbor Density Estimates", Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions , 36(5):1139–1148.

[20] Ozger, Z.B. and Amasyali, M.F., (2013). "KNN Parameter Selection Via Meta Learning", in Signal Processing and Communications Applications Conference (SIU), 24-26 April 2013, Trabzon Turkey.

[21] Sanchez, J.S., Pla, F. and Ferri, F.J., (1997). "On the use of neighbourhood-based non-parametric classifiers”, Elsevier Pattern Recognition Letters, 18(11-13):1179–1186.

[22] Ghosh, A.K., (2007). "On Nearest Neighbor Classification Using Adaptive Choice of k”, Journal of Computational and Graphical Statistics, 16(2):482-502.

[23] Guo, G., Wang, H., Bell, D., Bi, Y. and Greer, K., (2003). "KNN Model-Based Approach in Classification”, Lecture Notes in Computer Science, (2888):986-996.

[24] Fürnkranz, J. and Petrak, J., (2001). "An evaluation of landmarking variants”, Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning, 57-68.

[25] Martin, E., Hans-Peter, K., Jörg, S. and Xiaowei, X., (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise”, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), 226-231.

[26] Simon, P., (2013). “Too Big to Ignore: The Business Case for Big Data”, 89-100; Wiley Press, New York.

[27] Wang, S., (2003). Artificial Neural Network, Interdisciplinary Computing in Java Programming., 81-100; Springer Press US, New York.

[28] Polikar, R., (2006). "Ensemble based systems in decision making”, IEEE Circuits and Systems Magazine, 6(3):21-45.

[29] Rokach L., (2010). "Ensemble-based classifiers”, Artificial Intelligence Review, 33(1-2):1-39.

[30] Zhou, Z., (2012). Ensemble Methods Foundation and Algorithms. Machine Learning and Patteren Recognition Series: CRC Press, Toylor Francis Group, A Chapman & Hill Book, New York.

[31] Dietterich, T., G., (1997). "Machine learning research: Four current directions.”, AI Magazine, 18(4):97–136.

[32] Rokach, L., (2010). Pattern Classification Using Ensemble Methods (Series in Machine Perception and Artifical Intelligence), World Scientific Publishing Company, Singapur.

[33] Sewell, M., (2008). "Ensemble Learning, Research Notes", UCL Department Of Computer Science, 1-16.

[34] Clark, P. and Boswell, R., (1991). "Rule induction with CN2: Some recent improvements", in In Proceedings of the European Working Session on Learning, Pitman, 151-163.

[35] Buntine, W., (1990). "A Theory of Learning Classification Rules. ", Doctoral dissertation, School of Computing Science, University of Technology., Sydney, Australia.

[36] Shilen, S., (1990). "Multiple binary tree classifiers", Pattern Recognition, 23(7):757-763.

[37] Jing, L., (2007). "An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data", Knowledge and Data Engineering, IEEE Transactions, 19(8):1026-1041.

[38] Tesauro, (1995). Advances in Neural Information Processing Systems 7, MIT Press, Denver, Colorado.

[39] Choi, S. and Sohn, Y., (2001). "Ensemble based on Data Envelopment Analysis", in ECML Meta Learning workshop, 4 Sep. 2001.

[40] Hansen, J., (2000). Combining Predictors. Meta Machine Learning Methods and Bias Variance & Ambiguity Decompositions. PhD dissertation. : Aurhus University.

[41] Ghosh, K. and Tumer, J., (2000). Robust Order Statistics based Ensembles for Disributed Data Mining. Advances in Distributed and Parallel Knowledge Discovery: AAAI/MIT Press, 185-210.

[42] Zenko, B. and Dzeroski, S., (2004). "Is Combining Classifiers with Stacking Better than Selecting the Best One?", Machine Learning, 54(3):255–273.

[43] Chan, S.J. and Stolfo, P.K., (1995). "A Comparative Evaluatin of Voting and Meta learning on Partitioned Data, Proc. ", in 12th Intl. Conf. On Machine Learning, ICML-95.

[44] Chan, P.K. and Stolfo, S.J., (1997). "On the Accuracy of Meta-learning for Scalable Data Mining", J. Intelligent Information Systems, , 8:5-28.

[45] Furnkranz, A.K. and Seewald, J., (2001).Grading classifiers.: Austrian research

institute for Artificial intelligence.

[46] Rokach, L., (2010). Ensemble Methods in Supervised Learning, 959-979; Springer Press US, New York.

[47] BREIMAN, L., (1996). Bagging Predictors, Machine Learning, 123–140; Kluwer Academic Publishers, Boston.

[48] Schapire, R.E., (1990). "The strength of weak learnability", Machine Learning, 5(2):197–227.

[49] Schapire, Y. and Freund, R.E., (1996). "In: Lorenza SAITTA, Machine Learning: Proceed ings of the Thirteenth International Conference (ICML '96)", 148–156; Experiments with a New Boosting Algorithm, San Francisco.

[50] Kuncheva, L.I. and Alonso, C.J., (2006). "Rotation Forest: A New Classifier Ensemble Method", Pattern Analysis and Machine Intelligence, IEEE Transactions, 28(10):1619 - 1630.

[51] Breiman, L., (2001). "Random forests", Machine Learning, 45(1):5-32.

[52] Wolpert, D.H., (1992). "Stacked Generalization", Neural Networks, 5(2):241-259.

[53] Ho, T.K., (1998). "The Random Subspace Method for Constructing Decision Forests. ", IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8):832-844.

[54] Kuncheva, L.I. and Rodriguez, J.J., (2007). "Classifier ensembles with a random linear oracle", Knowledge and Data Engineering IEEE Transactions, 19(4):500-508.

[55] Vilalta, R. and Youssef, D., (2002). "A perspective view and survey of meta-learning", Artificial Intelligence Review, 18(2):77-95.

[56] Browne, M.W., (2002). "Journal of Mathematical Psychology", Journal of Mathematical Psychology, Elsevier, 44(1):108-132.

[57] Do, K., Ambroise, C. and McLachlan, G.J., (2005). Analyzing Microarray Gene Expression Data, Wiley Series in Probability and Statistics, New York.

[58] Ho, T.K. and Basu, M., (2002). "Complexity Measures of Supervised Classification Problems", IEEE Transaction on Pattern Analysis and Machine

Intelligence, 24(3):289-300.

[59] Marcia, N., Ho, T.K. and Orriols-Puig, A., (2010). Documentation for the Data Complexity Library in C++, GRSI Report No.2010001, Barcelona, Spain.

[60] Malina, W., (2001). "Two-parameter fisher criterion", IEEE Transactions on Systems, Man, Cybernetics Part B : Cybernetics , 31(4):629–636.

[61] Moore, E.H., (1920). "On the reciprocal of the general algebraic matrix", Bulletin of the American Mathematical Society, 26(1):394–395.

[62] Penrose, R., (1955). "A generalized inverse for matrices.", In Proceedings of the Cambridge Philosophical Society, 51(1):406–413.

[63] Ho, T.K. and Baird, H.S., (1998). "Pattern Classification with Compact Distributions Maps", Computer Vision and Image Understanding, 70(1):101-110.

[64] Vapnik, V., (1995). "The nature of statistical learning theory", Springer Verlag, New York.

[65] Platt, J.C., (1998). Fast training of support vector machines using sequential minimal optimization, Advances in Kernel Methods: Support Vector Learning, pages 185–208; MIT Press, Cambridge MA, USA.

[66] Jones, R. and King, G.P., Broomhead, D.S., (1987). "Topological Dimension and Local Coordinates", J. Physics, A: Mathematical and General, 20(6):L563-L569.

[67] Smith, S.P. and Jain, A.K., (1988). "A Test to Determine the Multivariate Normality of a Data Set", IEEE Trans. Pattern Analysis and Machine Intelligence, 10(5):757-761.

[68] Verveer, P.J. and Duin, R.P.W., (1995). "An Evaluation of Intrinsic Dimensionality Estimators", IEEE Trans. Pattern Analysis and Machine Intelligence, 17(1):81-86.

[69] Wyse, N., Dubes, R. and Jain, A.K., (1980). "A Critical Evaluation of Intrinsic Dimensionality Algorithms", Pattern Recognition in Practice, E.S. Gelsema and L.N. Kanal, eds., North-Holland, 415-425.

[70] Ghosh A.K., (2006). "On optimum choice of k in nearest neighbor classification", Elsevier Computational Statistics & Data Analysis (CSDA),

50(11):3113 – 3123.

[71] A. Hoekstra and R. P. W. Duin., (1996). "On the nonlinearity of pattern classifiers.", in In Proceedings of the 13th International Conference on Pattern Recognition, 271–275; 25-29 Aug 1996, Washington DC, USA.

[72] F. Lebourgeois and H. Emptoz, (1996). "Pretopological approach for supervised learning.", in In Proceedings of the 13th International Conference on Pattern Recognition, IEEE Computer Society, 256–260; Washington DC, USA.

[73] Bache, K. and Lichman, M., (2013). UCI Machine Learning Repository http://archive.ics.uci.edu/ml, 29/10/2014.

[74] Alcal´a-Fdez, J., (2011). "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework.", Journal of Multiple-Valued Logic and Soft Computing, 17(2-3) 255-287.

[75] Witten I. and Frank, E., (2005). Data mining : Practical machine learning tools and techniques. San Francisco: Morgan Kaufmann.

[76] Willmott C.J. and Matsuura, K., (2005). "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance", Climate research, 30(1):79-82.

[77] Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M.K., (1987). "Occam's Razor", Elsevier Information Processing Letters, 24(6):377-380.

[78] Simon, H., (1998). Neural Networks: A Comprehensive Foundation, 40-100; 2nd Edition, Prentice Hall, New York.

[79] Hanley, J.A. and McNeil, B.J., (1982). "The meaning and use of the area under a receiver operating characteristic (ROC) curve", Radiology, 143(1):29-36.

[80] Matthews, B.W., (1975). "Comparison of the predicted and observed secondary structure of T4 phage lysozyme", Biochimica et Biophysica Acta (BBA) - Protein Structure, 405(2):442-451.

[81] Ho, T.K. and Basu, M., (2002). "Complexity Measures of Supervised Classification Problems", IEEE Transaction on Pattern Analysis and Machine Intelligence, 24(3):289-300.

[82] Dempster, A.P., Laird, N.M. and Rubin, D.B., (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm", Journal of the Royal Statistical Society, B(29):1-38.

[83] Dempster, A.P., Laird, N.M. and Rubin, D.B., (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm", Journal of the Royal Statistical Society, JSTOR 2984875. MR 0501537, B(39):1-38.

[84] Murphy, K.P., (2012). "Machine learning : a probabilistic perspective", 151-152; Cambridge: MIT Press, New York.

[85] Ester, M., Kriegel, H.P., Sander, K. and Xu, X., (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise", in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), 226–231; AAAI Press.

[86] Campello, R., Moulavi, D. and Zimek, A., (2013). "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies", Data Mining and Knowledge Discovery, 27(3):344.

[87] Xu, L., (1998). "RBF nets, Mixture Experts, and Bayesian Ying-Yang Learning", Elsevier Neurocomputing, 19: 223-257.

[88] Kubat, M., (1998)."Decision Trees Can Initialize Radial-Basis Function Networks", IEEE Transactıon On Neural Networks, 9(5):813-821.

[89] Zhi-Hua, Z., (2012). "Ensemble Methods: Foundations and Algorithms", in Ensemble Methods: Foundations and Algorithms, 270-272; CRC Press, Florida.

[90] Ian H.W. and Eibe, F., (2005). "Data Mining: Practical Machine Learning Tools and Techniques", Elsevier, 2: 196-198.

[91] Myoung, J.K., Min, S.H. and Han, I., (2006), "An evolutionary approach to the combination of multiple classifiers to predict a stock price index", Elsevier Expert Systems with Applications, 31 : 241–247.

[92] Shepard, D., (1968), "A Two-dimensional interpolation function for irregularly spaced data", 517-523; Proceedings of the 23rd National Conference of the ACM.

[93] Bache, K. and Lichman, M., (2013). UCI Machine Learning Repository, http://archive.ics.uci.edu/m, 08/06/2014.

[94] Demsar, J., (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets”, Journal of Machine Learning Research (JMLR), 7: 1-30.

[95] Gunopulos, D., Ma, S., Yan, B., Al-Razgan, M., Papadopoulos, D. and Domeniconi, C., (2007). "Locally adaptive metrics for clustering high dimensional data”, Data Mining and Knowledge Discovery, 14(1):63-97.

[96] Xuan, G., Zhang, W. and Chai, P., (2001). "EM algorithms of Gaussian mixture model and hidden Markov model”, in IEEE International Conference on Image Processing, 07-10 Oct 2001, Thessaloniki.

[97] Sanguinetti, G., (2005). "Automatic Determination of the Number of Clusters Using Spectral Algorithms”, Machine Learning for Signal Processing IEEE Workshop, 55-60.

[98] Brugger, D., Bogdan, M. and Rosenstiel, W., (2008). "Automatic Cluster Detection in Kohonen's SOM”, Neural Networks, IEEE Transactions, 19(3):442-459.

[99] Kohonen, T., (2001). “Self-organizing maps” 30-50; Springer Science & Business Media, New York.

[100] Amasyali, M.F. and Ersoy, O., (2009). "A Study of Meta Learning for Regression”, ECE Technical Report, Purdue University, http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1385&context=ecetr , 29 March 2014.

[101] Myat, G.J., (2007). A Practical Guide to Exploratory Data Analysis and Data Mining Making Sence of Data, 176-181; Wiley Publishing, New York.

[102] Bhatia, N., (2010). "Survey of Nearest Neighbor Techniques”, (IJCSIS) International Journal of Computer Science and Information Security, 8(2):302-305.

[103] Cai, Z., Wang, D., Jiang, S. and Jiang, L., (2007). "Survey of Improving K-Nearest-Neighbor for Classification “, Fuzzy Systems and Knowledge Discovery, Fourth International Conference, 679-683; 24-27 Aug. 2007, Haikou.

[104] MILOUD-AOUIDATE, A. and BABA-ALI, A.R., (2011). "Survey of Nearest Neighbor Condensing Techniques”, (IJACSA) International Journal of Advanced Computer Science and Applications, 2(11):59-64.

[105] Li, S. Z., (2008). "{N}earest feature line”, The Scholarpedia, 3(3):4357.

[106] Zhang, C., and Zhou, J.W.Y., (2004), "Tunable Nearest Neighbor Classifier”, Pattern Recognition, Lecture Notes in Computer Science, 204-211.

[107] McNames, J., (2001). "A fast nearest-neighbor algorithm based on a principal axis search tree”, Pattern Analysis and Machine Intelligence, IEEE Transactions, 23(9):964 - 976.

[108] Maw-Lin, Leou., Chien-Min, W. and Yi-Ching, L., (2010). "Fast exact k nearest neighbors search using an orthogonal search tree”, Pattern Recognition, 43(6):2351–2358.

[109] Agapitos, A., O’Neill, M. , and Brabazon A., (2013). "Adaptive Distance Metrics for Nearest Neighbour Classification Based on Genetic Programming”, Lecture Notes in omputer Science, Springer Berlin Heidelberg, 7831:1-12.

[110] Wang, J., Neskovic, P. and Cooper, L.N., (2007). "Improving nearest neighbor rule with a simple adaptive distance measure”, Elsevier Pattern Recognition Letters, 28(2):207-213.

[111] Fandos, R., Debes, C. and Zoubir, A.M., (2013). "Resampling methods for quality assessment of classifier performance and optimal number of features”, Elsevier Signal Processing , 93: 2956–2968.

[112] The MATLAB R2014a Tutorial - On-Line Document. [Online]. www.mathworks.com/help/stats/kdtreesearcher-class.html, 05/08/2014.

[113] Weiss, M.A., (2013). Data Structures & Algorithm Analysis in C++, 83-85, 614-618, 62; 4th Edition, Pearson, New York.

[114] Myatt, G.J., (2007). Making Sence of Data: A Practical Guide toExploratory Data Analysis and Data Mining, 120-129; Wiley Publishing, New York.

[115] Demsar, J., (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets”, Journal of Machine Learning Research (JMLR), 7(1):1-30.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname	: Faruk BULUT
Date of Birth and Place	: 01.28.1974, Kayseri
Foreign Language	: English
E-mail	: bulutfaruk@gmail.com

EDUCATION

Degree	**Department**	**School/University**	**Date of Graduation**
Master	Computer Engineering	Fatih University	2010
Undergraduate	Computer/Electronics Edu.	Marmara University	1998
High School	Computer	Ankara Gazi Teknik Lisesi	1992

PUBLISHMENTS

Submitted Journals (SCI-Expanded)

1. Faruk BULUT, M. Fatih AMASYALI, "A New Meta Learning Approach: Why Decision Trees Outperform or Fail on a Dataset?", Journal of Information Science and Engineering (JISE), ISSN 1016-2364, Manuscript No.: 140812, (on reviewing process)
2. Faruk BULUT, M. Fatih AMASYALI, "Locally Adaptive k Parameter Selection for Nearest Neighbor Classifier: One Nearest Cluster", Pattern Analysis and Applications (PAAA), ISSN: 1433-7541, Manuscript No.: PAAA-D-15-00061. (on reviewing process)

3. Faruk BULUT, M. Fatih AMASYALI,“A New Approach in Mixture of Experts using Hard Clustering”, Science China-Information Sciences (SCIS), ISSN: 1674-733X, Manuscript No.: SCIS-2015-0171. (on reviewing process)

National Journals

1. Faruk Bulut, M.Fatih Amasyalı, “Boosting the Performance of Instance Based Classifiers by Using Clustering”, “Örnek Tabanlı Sınıflandırıcılda Kümeleme Yöntemiyle Performans Artırımı”, Dokuz Eylül Üniversitesi Mühendislik Fakültesi Mühendislik Bilimleri Dergisi, Cilt 17, Sayı 52, No:4, ISSN: 1302-9304, 2015.
2. Faruk Bulut, M.Fatih Amasyalı, A. Coşkun Sönmez, “A New Approach in Mixture of Experts:Classification with Hard Clustering and a New Gate Function”, “Uzman Karışımlarında Yeni Bir Yaklaşım: Katı Kümeleme ve Yeni Bir Geçiş Fonksiyonuyla Sınıflandırma”, İzmir Katip Çelebi Üniversitesi Akıllı sistemler Dergisi, vol. 1, pp. 1-12., 2015.

Conference Papers

1. Faruk Bulut, M.Fatih Amasyalı, A. Coşkun Sönmez, "A New Gating Function for Mixture of Experts”, “Uzman Karışımlarında Yeni Bir Yaklaşım: Uzman Kararlarının Yeni Bir Geçiş Fonksiyonuyla Birleştirmesi”, Akıllı sistemlerde Yenilikler ve Uygulamaları (ASYU) 2014, October 9-10, İzmir, 2014.
2. Faruk Bulut, M.Fatih Amasyalı, "Sample Based Dynamic K selection in *k*-NN Algorithm", “En Yakın k Komşuluk Algoritmasında Örneklere Bağlı Dinamik k Seçimi”, Akıllı sistemlerde Yenilikler ve Uygulamaları (ASYU) 2014, October 9-10, İzmir, 2014.