FEATURE WEIGHTING PROBLEM IN K-NEAREST NEIGHBOR CLASSIFIER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

NURULLAH GÜLEÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE IN
OPERATIONAL RESEARCH

SEPTEMBER 2017

Approval of the thesis:

FEATURE WEIGHTING PROBLEM IN K-NEAREST NEIGHBOR CLASSIFIER

submitted by **NURULLAH GÜLEÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Operational Research Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Assoc. Prof. Dr. Cem İyigün
Head of Department, **Operational Research** _____

Assoc. Prof. Dr. Cem İyigin
Supervisor, **Industrial Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Nur Evin Özdemirel
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Cem İyigün
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Serhan Duran
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Ceylan Yozgatlıgil
Statistics Dept., METU _____

Assist. Prof. Dr. Ercüment Çiçek
Computer Engineering Dept., Bilkent University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: NURULLAH GÜLEÇ

Signature :

ABSTRACT

FEATURE WEIGHTING PROBLEM IN k-NEAREST NEIGHBOR CLASSIFIER

Güleç, Nurullah

M.S., Department of Operational Research

Supervisor : Assoc. Prof. Dr. Cem İyigün

September 2017, 107 Pages

The *k*-Nearest Neighbor (*k*-NN) algorithm is one of the well-known and most common used algorithms for the classification problems. In this study, we have focused on feature weighted *k*-NN problems. Two different problems are studied. In the first problem, *k* value and the weights of each feature are optimized to maximize the classification accuracy. Objective function of the problem is nonconvex and nonsmooth. As a solution approach, Forest Optimization Algorithm (FOA), which is a newly introduced evolutionary algorithm, has been considered. Two different algorithms based on FOA are proposed. In the latter problem, class dependency on the feature weights is considered and class dependent feature weighted *k*-NN problem is studied where the feature weights are different for each class for maximizing the classification accuracy. A solution algorithm again based on FOA is proposed. All proposed algorithms are tested on different benchmark data sets and the numerical results are reported. Performances of the algorithms are also compared with the other algorithms from the other studies in the literature.

Keyword: *k*-Nearest Neighbor Classifier, Forest Optimization Algorithm, Feature Weighted *k*-NN, Class Dependent Feature Weighted *k*-NN

ÖZ

K-EN YAKIN KOMŞU SINIFLANDIRMA ALGORİTMASINDA ÖZELLİK AĞIRLIKLANDIRMA PROBLEMİ

Güleç, Nurullah

Yüksek Lisans, Yöneylem Araştırması Bölümü

Tez Yöneticisi : Doç. Dr. Cem İyigün

Eylül 2017, 107 Sayfa

K-En Yakın Komşu (*k*-NN) algoritması sınıflandırma problemleri için iyi bilinen ve en çok kullanılan algoritmalardan biridir. Bu çalışmada özellik ağırlıklı *k*-NN problemi üzerinde duruldu. İki farklı problem çalışıldı. İlk problemde, *k* değeri ve özelliklerin ağırlıkları sınıflandırma doğruluğunu maksimize etmek için optimize edildi. Problemin amaç fonksiyonu konveks olmayan bir yapıdadır. Çözüm yaklaşımı olarak, yeni taktim edilen bir gelişim algoritması olan Orman Optimizasyon Algoritması (FOA) düşünüldü. FOA'ya dayanan iki farklı algoritma önerildi. Sonraki problemde, özellik ağırlıklarının sınıfa bağlılığı göz önünde bulunduruldu ve sınıflandırma doğruluğunu maksimize etmek için her sınıf ait özellik ağırlıklarının farklı olduğu, sınıfa bağlı özellik ağırlıklı *k*-NN problemi çalışıldı. Çözüm algoritması yine FOA'ya dayalı olarak önerildi. Önerilen tüm algoritmalar farklı kıyaslama data setleri üzerinde test edildi ve sayısal sonuçlar rapor edildi. Algoritmaların performansı litaratürde diğer çalışmalardaki algoritmalarla da kıyaslandı.

Anahtar Kelimeler: *k*-En Yakın Komşu Sınıflandırma Algoritması, Orman Optimizasyon Algoritması, Özellik Ağırlıklı *k*-NN, Sınıfa Bağımlı Özellik Ağırlıklı *k*-NN

To My Wife and little Son

ACKNOWLEDGMENTS

First of all, I wish to present my most sincere gratitude to my thesis advisor Assoc. Prof. Dr. Cem İyigün for his continuous support, encouragement and motivating me with sincere attitude. I have to faithfully express that without his wise consultancy and sincere attitude, this work could not have been completed successfully.

I am grateful to The Scientific and Technological Research Council of Turkey (TÜBİTAK) for financial support within 2211-Domestic Graduate Scholarship Program.

I would like to thank my parents for their financial and spiritual support throughout my education life. I know their prayers are always with me.

Finally, I give my greatest thanks to my wife Merve GÜLEÇ and my little son Mehmet Fatih GÜLEÇ for their unlimited support to me, although I had neglected them. Thank you endlessly.

TABLE OF CONTENTS

ABSTRACT vi
ÖZ vii
ACKNOWLEDGMENTS ix
TABLE OF CONTENTS x
LIST OF TABLES xiii
LIST OF FIGURES xv
LIST OF ABBREVIATIONS xvii
CHAPTERS
1. INTRODUCTION 1
2. BACKGROUND AND LITERATURE REVIEW 7
2.1. Terminological Background 7
2.1.1. Data and Data Set 7
2.1.2. Machine Learning 9
2.1.3. Supervised Learning 11
2.1.4. Unsupervised Learning 11
2.1.5. Reinforcement Learning 11
2.1.6. Separation Of Data Set and Cross Validation 12
2.1.7. Classification 15
2.1.8. Feature Weight 15
2.1.9. Distance Measure 15
2.1.10. Accuracy 16
2.2. Literature Review 16
2.2.1. Nearest Neighbor 17
2.2.2. *k*-Nearest Neighbor Algorithm 18
2.2.3. Weighted *k*-Nearest Neighbor Algorithm 25
2.2.4. Modified *k*-Nearest Neighbor 25

2.2.5. Feature Weighted *k*-Nearest Neighbor 27
3. FOREST OPTIMIZATION ALGORITHM 31
3.1. Life Cycle Of Trees 31
3.2. The Forest Optimization Algorithm's Inspiration From The Life Cycle of Trees 33
3.3. Parameters of the Forest Optimization Algorithm 35
3.4. Steps of the Forest Optimization Algorithm 36
3.4.1. Creating Initial Trees 36
3.4.2. Local Seeding 37
3.4.3. Population Limitation 39
3.4.4. Global Seeding 39
3.4.5. Ranking the Trees 40
3.4.6. Stopping Condition 40
3.5. Testing Forest Optimization Algorithm 42
3.5.1. Some Examples From Test Function 44
3.5.2. All Results in Test Functions 49
4. FEATURE WEIGHTED *k*-NEAREST NEIGHBOR PROBLEM 53
4.1. Algorithm 1 56
4.1.1. Implementation Details of Algorithm 1 61
4.2. Algorithm 2 62
4.3. Application Results 66
4.3.1. Results of Algorithm 1 67
4.3.2. Results of Algorithm 2 72
4.3.3. Comparison of the Results of Two Proposed Algorithms 74
5. CLASS DEPENDENT FEATURE WEIGHTED *k*-NN 77
5.1. Feature Weights 79
5.2. Distance Measure 80
5.3. Forest Optimization Algorithm Application 83
5.4. Results of Algorithm 3 84
6. CONCLUSION 89

REFERENCES 93
APPENDICES
APPENDIX A 99
APPENDIX B 107

LIST OF TABLES

Table 1: Data Set 8
Table 2: Normalized Data Set 9
Table 3: Synthetic Data Set 22
Table 4: Distance between Y and Training Data 24
Table 5: *k*-Nearest Neighbors of Y 24
Table 6: Calculated Distances with weight set in Eq.14 29
Table 7: Calculated Distances with weight set in Eq. 16 30
Table 8: Pseudo-Code of Forest Optimization Algorithm 41
Table 9: Test Functions 43
Table 10: FOA Parameters for Test Functions 44
Table 11: Results of FOA 50
Table 12: Brief Presentation of the Algorithm 1 58
Table 13: Pseudo-code of Algorithm 1 60
Table 14: Pseudo-code of Algorithm 2 65
Table 15: Properties of Data Sets 66
Table 16: FOA Parameters 67
Table 17: Best *k* neighbor values at end of the first step 67
Table 18: Classification Accuracy at the end of the second step 68
Table 19: Best *k* neighbor values at end of the second solution of Subproblem 1 69
Table 20: Classification Accuracy at the end of the fourth step 70
Table 21: Best Classification Accuracy at the end of Algorithm 1 71
Table 22: Algorithm 1 vs. Some Other Algorithms 71
Table 23: Best Classification Accuracy at the end of Algorithm 2 73
Table 24: Comparison of Two Proposed Algorithms 74
Table 25: Algorithm 1 vs. Algorithm 2 75
Table 26: Parameters of FOA for CDFW *k*-NN 84

Table 27: Best Classification Accuracy of Algorithm 3 85
Table 28: CDFW *k*-NN vs FW *k*-NN 85
Table 29: Sample Weight Set for IRIS Data Set 86
Table 30: Performance of Alg.1, Alg.2 and Alg.3 87

LIST OF FIGURES

Figure 1: Machine Learning 10
Figure 2: The Holdout Cross-Validation Method 13
Figure 3: The 5-Fold Cross-Validation Method 13
Figure 4: The Monte Carlo Cross-Validation Method 14
Figure 5: Toy Example: NN 18
Figure 6: Toy Example: *k*-NN 23
Figure 7: Flowchart of Forest Optimization Algorithm 35
Figure 8: Local Seeding, LSC=5 38
Figure 9: Local Seeding, *LSC*=2 and Random Numbers : [-0.2 0.7] 38
Figure 10: Global Seeding Example with *GSC*=2 40
Figure 11: 3D Plot of F1 Function 45
Figure 12: Results of FOA for F1 Function 46
Figure 13: Fitness Value of Best and Worst Tree in Population and Mean of Population for F1 46
Figure 14: 3D Plot of F6 Function 47
Figure 15: Results of FOA for F6 Function 48
Figure 16: Fitness Value of Best and Worst Tree in Population and Mean of Population for F6 48
Figure 17: 3D Graph of Classification Accuracy Function 55
Figure 18: Flowchart of the Algorithm 1 59
Figure 19: Flowchart of the Algorithm 2 63
Figure 20: First Tree of Initial Forest of BUPA Data Set 72
Figure 21: Toy Example 78
Figure 22: Selected Some Points from Toy Example in Figure 21 81
Figure 23: 3D graph of F1 99

Figure 24: 3D graph of F2 99
Figure 25: 3D graph of F3 100
Figure 26: 3D graph of F4 100
Figure 27: 3D graph of F5 101
Figure 28: 3D graph of F6 101
Figure 29: 3D graph of F7 102
Figure 30: 3D graph of F8 102
Figure 31: 3D graph of F9 103
Figure 32: 3D graph of F10 103
Figure 33: 3D graph of F11 104
Figure 34: 3D graph of F12 104
Figure 35: 3D graph of F13 105
Figure 36: 3D graph of F14 105
Figure 37: 3D graph of F15 106

LIST OF ABBREVIATIONS

CDFW*k*NN	Class Dependent Feature Weighted *k*-Nearest Neighbor
FOA	Forest Optimization Algorithm
FW*k*NN	Feature Weighted *k*-Nearest Neighbor
GSC	Global Seeding Changes
k-NN	*K*-Nearest Neighbor
LSC	Local Seeding Changes
M*k*NN	Modified k-Nearest Neighbor
NN	Nearest Neighbor

CHAPTER 1

INTRODUCTION

The rapid technological development in the world and the ease of people's access to technology have led to significant increases in the amount of data produced, especially since 2000's. Along with the development and widespread use of the internet, all tools used on the internet provide an important contribution to the increase of these data. From mobile phones we use every day in daily life to devices used in hospitals, from enterprise resource planning software used in companies, to studies done at universities, large amounts of data are produced and recorded every day. It is estimated that the amount of data to be produced in the next ten years will be about ten times higher than the data which produced from the formation of the World to today. Besides this, the quantity and variety of goods and services produced in the world are increasing due to the increasing consumption demands of people day by day. This increase in the amount of production makes the production mechanization obligatory. It is thought that the machinery in production increases both the production amount and standardization in production.

The increasing amount of data produced in the world and the need to mechanized production have made it impossible to analyze data in a conventional way. In order to solve these and similar needs through machines, different methods have been developed and continue to be developed new methods every day. These methods are called as machine learning which can be defined, in the simplest definition, that machines can make decisions by analyzing situations and all of

the algorithms used to provide them this ability is called machine learning algorithms. Machine learning is programming of computer to optimize the previously defined performance criteria by analyzing data from examples or past experiences (Alpaydın, 2014).

Machine learning algorithms are used today in many areas we are not aware such friendship advice in social media, risk analysis in the finance sector, customer expectation analysis in marketing, optimal route advice on navigation. Some different recent studies which machine learning algorithms used are bankruptcy prediction and credit risk in Barboza et al. (2017), material science in Yue et al. (2017), rainfall prediction in Cramer et al. (2017) and medical images analysis in Criminisi (2016).

The two most fundamental areas that machine learning algorithms seek to solve are clustering and classification problems. Clustering can be expressed as the problem of dividing instance set, according to their similar properties, in clusters which are not predefined. Clustering, in another definition, is the problem of searching for a pattern among the samples in the data set. In classification problems, unlike clustering methods, the classes to which the samples belong are already known. Classification is an assignment problem. The main purpose of the classification problem is to create a model by using instances in the training set, classes they belong is known, and to classify correctly the instances to be obtained later. As it can be understood from these definitions, clustering problems can be called as unsupervised classification problems.

K-means is most commonly used algorithm for clustering problems. In the *k*-means algorithm, the goal is to divide the data set consisting of n samples into *k* clusters. When the samples are divided into clusters, it is targeted that the samples in the same cluster have similar properties and the samples in different clusters have different properties. The similarity between samples is determined by calculating distances. The most commonly used method of distance

determination is the Euclidean distance. The nearest instances based on calculated distance are considered to be more similar to each other in this method.

The most commonly used algorithm for solving classification problems is the *k*-Nearest Neighbor(*k*-NN) method. While a sample appoints in the *k*-NN method, the nearest *k* sample in the training set to that sample is taken into account. The sample which class is undefined is assigned the class which has maximum member in the selected *k*-nearest sample. Different measurements are used in determining the closeness in the *k*-NN algorithm such as Euclidean Distance, Minkowski Distance, Manhattan Distance, etc. The most commonly preferred distance measured is Euclidean distance. In summary, the basis of the *k*-NN algorithm is the *k* value and distance calculation. Therefore, choosing the *k* value and most appropriate distance measure are very significant issues for *k*-NN.

Evaluating all features with the same importance degree while distance is calculated in algorithms used in both classification and clustering studies is not an appropriate approach to real life problems. In classification and clustering algorithms, feature weighting studies have been implemented for a long time to achieve more successful results. Feature weighting applications in clustering studies have been ongoing for over 40 years (de Amorim, R. C., 2016). Some studies of feature weighting in clustering can be listed as DeSarbo et al. (1984), Modha and Spangler (2003), Chan et al. (2004), Huang et al. (2005). Feature weighting studies are widely used in classification problems. Some studies of feature weighting in classification can also be listed as Kira and Rendell (1992), Paredes and Vidal (2006), Sun (2007), Jiang et al. (2016) (Dialameh, M., & Jahromi, M. Z. (2017).

The feature weighting problem has a continuous and non-linear structure. This means that there are a lot of local extreme points and NP-Hard problems that can not be solved with mathematical methods. Feature weighting algorithms can be

considered as combinatorial and optimization problems and evolutionary algorithms for solving such problems are frequently suggested in the literature (Triguero, I. et al., 2012). Evolutionary algorithms are algorithms that are obtained by simulating life in nature as a computer program to solve complex problems. Genetic Algorithm, Ant Colony Optimization and Particle Swarm Optimization Algorithm are the most commonly used algorithms to solve problems in the literature.

In this study, feature weighted *k*-NN algorithm is worked on and two different problems are discussed. Feature weighted *k*-NN has two fundamental parameters, *k* neighbor values and feature weight set, **w**. These two parameters seriously affect the classification accuracy. For this reason, in the first problem, two different algorithms have been proposed to optimize the *k* value and feature weight set in this study. First proposed algorithm iteratively optimizes the *k* value and feature weight set. In the second algorithm, two parameters are optimized simultaneously. In both proposed algorithms, the main purpose is to maximize the classification accuracy by optimizing the *k* value and weights of the features. This problem based on *k* and **w** parameters has a nonlinear structure where it has many local minimums and maximums. Therefore, in order to optimize the *k* and **w** parameters in both proposed algorithms, an evolutionary algorithm Forest Optimization Algorithm (FOA) that is newly proposed in the literature by Ghaemi, M., & Feizi-Derakhshi, M. R., 2014, has used and implemented into the proposed algorithms.

The feature weights are assumed to be the same for all classes in the previous problem. In the second problem, feature weights are class-dependent allowing that each feature can have a different weight for each class. This problem again has two parameters, *k* and **w** parameters, like the previous problem and it has also a nonconvex and nonsmooth objective function. FOA is also used for developing an algorithm to solve the problem.

The outline of the thesis is as follows. In Section 2, there are the explanations of terms used in the study and the literature review about the methods used in the study. Section 3 is designed for description of FOA and the application of FOA on test functions. Fifteen test functions used in the literature were selected according to their properties and each was solved thirty times using FOA. Obtained results and resolution times are reported. In section 4, feature weighted *k*-NN method is applied using two different algorithms. In section 5, a class-based feature weighting algorithm is proposed and applied to data sets. In the final part, the results are discussed.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter is designed to explain the terms used to better understand the study and to summarize the researches related to this work in literature. In this context, firstly the terms used briefly will be defined and then examples will be given from the studies in the literature.

2.1. Terminological Background

In this study, the classification problem, which is one of the machine learning problems, has been studied. In this section of the study, to provide a better understanding of the methods used in subsequent chapters and to better explain the recommended algorithms, brief descriptions of the terms and methods used in the study are discoursed.

2.1.1. Data and Data Set

The fundamental definition of data is the unprocessed numerical or verbal values. Data can be obtained by using different research methods such as measurement, observation or experiment etc. Data is the basic values to obtain meaningful information by processing or analyzing.

Data set is a collection of data obtained by getting together the data collected about the same subject. Data sets are very important since they are one of the basic arguments used in scientific research. Data sets are usually stored as tables.

Table 1: Data Set

Name	Gender	Height (m)	Weight (kg)
Person 1	Woman	1.63	54.3
Person 2	Man	1.76	84.6
Person 3	Man	1.83	93.2
Person 4	Woman	1.70	59.8
Person 5	Woman	1.59	56.1

In Table 1, a sample data set is presented. This data set includes the gender, height and weight of 5 people in a group. All the values belonging to each person in Table 1 are data and the data set is formed by collecting these data in the same table.

Information is obtained by processing or analyzing the collected data. For example, the data set in Table 1 can be used to calculate the average of men's weights by 88.9 kg or the average height of women by 1.64 m. In this way, meaningful information can be obtained by processing the data.

In this study, each row in the data set will be named as a sample and each column will be named as feature. In data sets used for classification applications, one of the feature columns indicates the classes to which the samples belong. If the sample data set in Table 1 is looked, the gender of the samples can be thought of as the classes to which they belong. In this case, the data set consists 2 features, height and weight, and class of 5 samples.

In the next step, while analyzing the data set, it is seen that each feature has value in different intervals. This means that different features have different coefficients of effect during the data analysis stage and it can lead to incorrect results. To avoid this situation, data normalization is performed.

$$feature_i^{norm} = \frac{feature_i}{\max(feature_i)} \quad (1)$$

Data normalization is applied by dividing each feature by the maximum value of its interval and it is shown in Equation (1). In Table 2, the normalized version of sample data set is presented.

Table 2: Normalized Data Set

Sample	Class	Feature 1	Feature 2
Name	Gender	Height	Weight
Person 1	Woman	0.89	0.58
Person 2	Man	0.96	0.91
Person 3	Man	1.00	1.00
Person 4	Woman	0.93	0.64
Person 5	Woman	0.87	0.60

For normalization, all the data in the height column and all the data in the weight column are divided by 1.83 and 93.2, respectively. With normalization, the data of different features are standardized.

2.1.2. Machine Learning

Machine learning is algorithm that produces appropriate results by analyzing different and not encountered situation or data instead of algorithms that perform certain operations with standard codes. The purpose of using these algorithms is to make estimations and make decisions in previously inexperienced subjects by analyzing the old experiences or data.

The basis of determining the learning method to be used in machine learning is the characteristics of data to be learned. For example, in Table 1, if the gender is evaluated as a label for the samples, the application to be performed on this data may be to estimate the genders of new samples, which is not member of data set

now, according to the features. On the other hand, for unlabeled data, the main topic of the analysis is to determine the relationship between the data or to cluster the data. As it can be clearly understood that whether the data is labeled or unlabeled directly affects the work to be done and the learning method.

Machine learning algorithms can be examined in 3 basic categories according to the characteristics of the data to be learned: supervised learning, unsupervised learning and reinforcement learning. Machine learning methods and some algorithms that work with these methods are presented in Figure

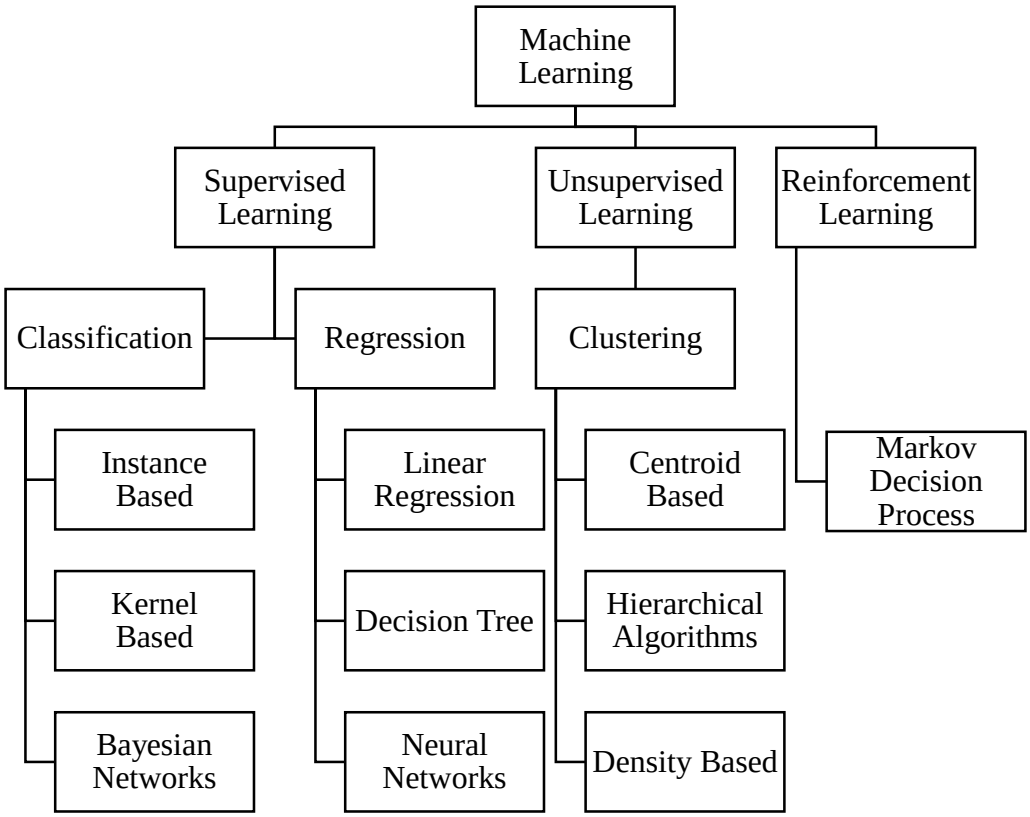

Figure 1: Machine Learning

2.1.3. Supervised Learning

Supervised learning is a method used in labeled data set to predict label of new data which don't consist in the data set. The purpose of supervised learning is to increase the accuracy of the results of the model used. In labeled date, since the samples are evaluated according to the labels, the learning outputs expected from the model are known. Thus, learning levels of the models trained can be determined. For this aim, the data set for supervised learning method is divided into 3 parts. These parts are the training set, the validation set and the test set.

2.1.4. Unsupervised Learning

Unsupervised learning is the study of the grouping of samples according to their similar characteristics, discovering hidden relationships between samples or similarities. In unsupervised learning applications, the data used are not labeled and therefore expectation from models is to discover clusters or hidden association rules between data as learning outcomes.

The most common unsupervised learning method is clustering analysis. One of the most critical issues is to decide how many clusters will be in the clustering analysis since data used is unlabeled. Another critical point is to define the similarity between the data.

2.1.5. Reinforcement Learning

In the reinforcement learning method, learning is occurred through the reward according to the satisfaction level of the outputs given by the algorithm. Unlike supervised learning, the correct input-output combinations in this method are not given to the algorithm. This method is used in situations where dynamic environmental conditions exist, such as games, scheduling, etc.

2.1.6. Separation Of Data Set and Cross Validation

Training and testing are necessary to measure the learning success of the algorithms. The use of all data in the hand during training causes problems during the testing of the algorithm. The purpose of the separation of the data set is training, choosing the most successful model and testing the selected model. The data set is divided into three parts as training set, validation set and test set. First, selected models are trained using the training set. After the training is completed, all models are compared with the learning levels using the validation set and finally, the best model in validation step is tested using the test set.

In some studies, the supervised learning method can be applied only by dividing into data set training set and test set since there is a small number of data in data set. Furthermore, the division of the data set by 3 reduces the training quality of the models as it reduces the number of samples in the training set. Because of such reasons, dividing the data set into two parts as training set and test set is called cross-validation.

In Gutierrez-Osuna, R. (2002), holdout cross-validation, *k*-fold cross-validation and leave one out cross-validation are presented, in addition to these, Dubitzky, W. et al (2012) also mention Monte Carlo cross-validation.

Holdout cross-validation is the most basic data set shredding method. In this method, the data set is divided into two parts as a training set and a test set as shown in Figure 2. Holdout cross-validation has two drawbacks. The model may not be trained sufficiently well because of the samples aside for the test set. Secondly, learning performance can show very high variations for different training and test sets.

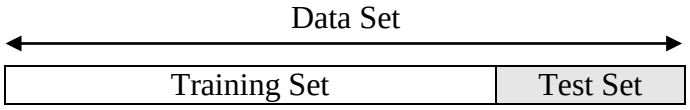

Figure 2: The Holdout Cross-Validation Method

The other method obtained by developing Holdout cross-validation is the *k*-fold cross-validation method. In this method, the data set is first divided into *k* pieces, and the holdout method is repeated *k* times so that every *k* pieces are the test set and the remaining *k*-1 pieces are the training set. The average of the errors calculated for each *k* test set is considered to be the error of the model.

In Figure 3, the example of *k*-fold cross-validation method for $k = 5$ is shown. As clearly shown in the example, the data set was divided into 5 equal parts and 5 training rounds were applied. In each round, one part of data set is used as a test set and errors are calculated.

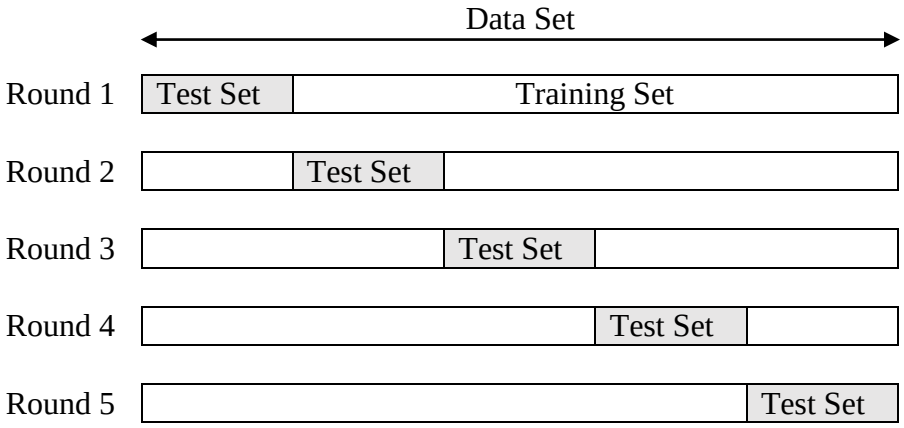

Figure 3: The 5-Fold Cross-Validation Method

The error of the training process of model is calculated as shown in Equation (2).

$$E = \frac{\sum_{i=1}^{k} e_i}{k} \tag{2}$$

Leave-one-out cross validation is another cross validation method. This method is a special application of *k*-fold cross validation at *k* = N (number of sample in data set). A sample is assigned a test set and the remaining samples are assigned as a training set. This assignment process repeats the number of samples times in the data set. As a result, error of the model is calculated as in Equation (3).

$$E = \frac{\sum_{i=1}^{N} e_i}{N} \tag{3}$$

Finally, in the Monte Carlo cross-validation method, also called randomized sub-sampling validation, the data set is divided randomly into two parts as test and training set. The model is trained by using training set and tested by using the test set. How many times this process will repeat is determined as a parameter. Unlike *k*-fold cross validation, in this model, an example can be found more than once in the test set, however, may not be found in test set.

Figure 4: The Monte Carlo Cross-Validation Method

An example of the Monte Carlo cross-validation model is presented in Figure 4. The test and training sets are selected randomly 5 times and the training errors of the model are calculated each time. Finally, total error of model is calculated as in Equation (2).

2.1.7. Classification

Classification is the study of categorizing the data into subgroups according to similarities of the data. Labeled data is used in classification studies. The main purpose of the classification study is to train the model using the labeled data at hand and to classify the new data correctly

2.1.8. Feature Weight

As shown in Table 1, samples in data set often have more than one feature. It is very likely that different features will have different significance or relevance level to achieve the desired result while working with data with multiple features. The variables used to take into account the different relevance levels of the features in the learn and test phase are called feature weights.

2.1.9. Distance Measure

Distance measure is used to mathematically express the similarity or dissimilarity between data. As the similarity between data increases, it is expected that the data are closer to each other. The Euclidean distance measure is widely used as distance measure in literature. Calculation of Euclidean distance measure is presented in Equation (4).

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{f=1}^{m} (a_i - b_i)^2} \quad (4)$$

In Equation (4), a_i and m represents the feature i of data **a**, number of features, respectively, and total distance is calculated by the square root of the sum of the squares of the differences between the features.

2.1.10. Accuracy

Accuracy shows what level the models which have completed the training process get the desired results in the test step. If an example is given from classification algorithms, accuracy is the ratio of correct classified data in the test set to the total number of data in the test set. In Equation (5), the accuracy calculation formula is shown.

$$Accuracy = \frac{number\ of\ correct\ classified\ data\ in\ test\ set}{number\ of\ all\ data\ in\ test\ set} \quad (5)$$

2.2. Literature Review

In this section of the study, the nearest neighbor (NN) classifier algorithm used in this study and the studies done in the literature will be summarized. The different methods applied to improve the performance of the NN algorithm and the feature weighted *k*-NN problem used in this study will be discussed.

2.2.1. Nearest Neighbor

In the literature, NN rules were first proposed by Cover, T. and Hart, P. (1967). In this study, NN is used as an effective pattern classification algorithm and assigning the data in the test set to the class is performed. Since the NN algorithm is easy to implement and has successful results, it has been used frequently and many improvements have been made on NN. The NN algorithm has been used in many field studies such as pattern recognition, object recognition and text mining (Bhatia, N., 2010).

The NN algorithm works on a very simple principle, such as assigning the data in the test set to class of its most similar data in the training set. The similarity between data is determined by the selected distance measure function. The most frequently used distance measure is the Euclidean distance measure function, given in Equation (4), to determine the similarity between data.

The NN algorithm can be understood more clearly, with an example. For example, there are 2 classes, each class consists of 4 samples in training set and one sample whose class is unknown. Samples in training set are shown in 3 dimensions; $\mathbf{x_1}(0.5,1,1), \mathbf{x_2}(1,0.5,1), \mathbf{x_3}(1,1.5,1), \mathbf{x_4}(1.5,1,1), \mathbf{x_5}(2.5,3,2)$, $\mathbf{x_6}(3,2.5,2)$, $\mathbf{x_7}(3,3.5,2)$ and $\mathbf{x_8}(3.5,3,1)$. The first two dimensions represent the features of the samples and the third dimension represents the class to which the samples belong. The scatter of these samples in the 2D plane is shown in Figure 5.

A sample $\mathbf{a}(1.4,2.5,?)$ whose class is unknown is shown in red square in Figure 5. While the NN algorithm is used, the distances between this sample and all samples in training set are calculated. In this example, 8 distances must be calculated by using Euclidean distance function. After all the distances have been calculated, sample whose class is unknown is assigned to the class of the

nearest point to itself. The closest sample of sample whose class in unknown is $\mathbf{x_3}(1,1.5,1)$, and the calculation of this distance is shown in Equation (6).

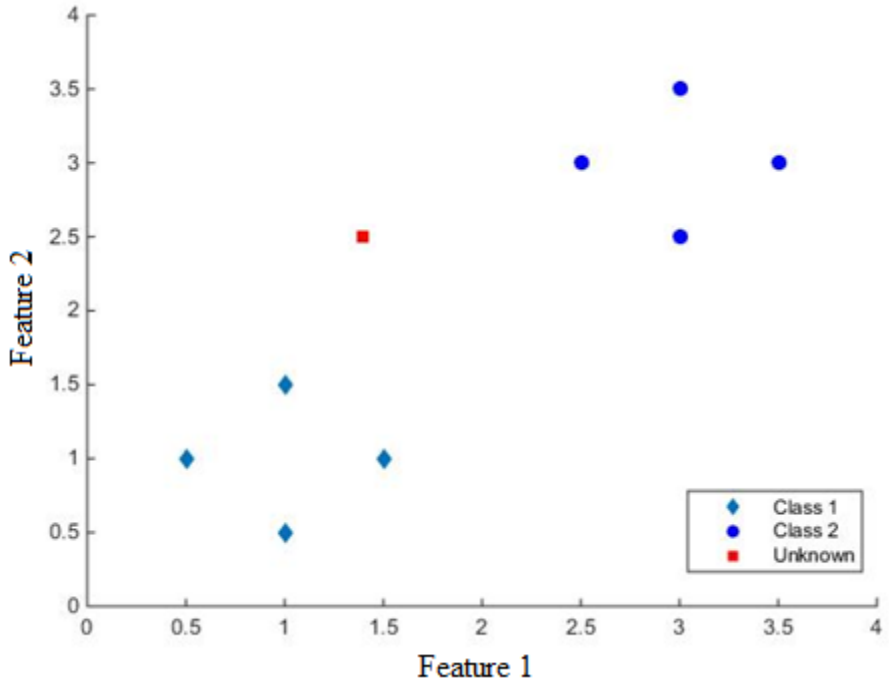

Figure 5: Toy Example: NN

$$d(\mathbf{a}, \mathbf{x_3}) = \sqrt{(1.4-1)^2 + (2.5-1.5)^2} \cong 1.077 \qquad (6)$$

As a result, the sample represented by data a is assigned to class 1.

2.2.2. *k*-Nearest Neighbor Algorithm

The *k*-Nearest Neighbor(*k*-NN) algorithm is an algorithm implemented by changing a parameter of the NN algorithm. In the NN algorithm, the closest neighbors are controlled while the class of data, whose class is unknown, is determined. However, in the *k*-NN algorithm, while the class of data is

determined, the nearest *k* neighbors are checked and classifications are made according to the class in which majority of neighbors are included.

The application of the *k*-NN algorithm is simple and *k*-NN can be applied successfully to data sets from different fields. Because of these properties, the *k*-NN algorithm has been used in many different problems from different fields in the literature and many improvements have been made on *k*-NN.

There are many studies on the feature weighting and feature selection problems that are used to improve the classification performance of the *k*-NN algorithm. In Byers, W. A. and Perone, S. P. (1980), feature weighting and feature selection methods were used at the same time in order to increase the classification accuracy and the classification success was improved by selecting the features according to the obtained weights. In other study, in Forbes R. A. (1986), it is aimed to increase the classification accuracy by using weights calculated by using the samples in the training set and proposed model was tested on chemical ionization data.

In the *k*-NN algorithm, there are very important parameters such as *k* neighbor value, distance measure, training set, etc. which are decided by the implementer and affect the final result directly. Many studies have been made in the literature for the optimization of these parameters. Some examples of these studies can be listed as follows. In Paliwal, K. K. and Rao, P. V. S. (1983), the *k*-NN algorithm was applied in the Vowel problem and the effect of different *k* values, training set sizes and distance measures on the classification accuracy was tested. Goin, J. E. (1984) stated that there are some deficiencies of the *k*-NN algorithm when data set has a small number of classes and each class has different number of samples, and In the 2-class *k*-NN, the result of the algorithm is directly affected by the *k* value and the number of samples. Keller, J. M. et al. (1985) have proposed a fuzzy k-nearest neighbor algorithm in which each sample in the training set does not have the same significance level in the classification

process. The fuzzy sets have been proposed to increase the classification accuracy and the fuzzy sets were assigned to the samples in the training set with 3 different approaches and the results were discussed. In Hattori, K. and Takahashi, M. (1999), *k*-NN was applied with a different perspective. For the samples in the test set, the closest *k* neighbors were selected from each class, and the weighted *k*-NN method was used to classify samples. The proposed model is sampled in a 2-class data set and reported to yield successful results.

The distance measure has always been an important parameter for the *k*-NN algorithm and many studies have been done. Todeschini, R. (1989) was analyzed how the use of different distance functions instead of the Euclidean distance function which is widely used as the distance measure in the literature, will affect the classification results. 6 different distance measures were applied to 4 different data sets and the results were compared and it was determined that different distance measure directly affected the classification accuracy. Belkasim, S. O. et al. (1992) stated that the calculation of distances takes the most time in the *k*-NN algorithm. For this reason, in this study, a method has been proposed that significantly reduces the distances needed to be computed in the *k*-NN algorithm. In proposed method, while the total distance calculated is reduced, the classification accuracy does not change. In Weinberger, K. Q. et al. (2006), to increase the classification accuracy, Mahalanobis distance function was used as distance measure and they aim that *k* neighbors belong always to the same class. Wang, J. et al. (2007) proposed a simple adaptive distance measure to improve the classification performance.

The *k*-NN algorithm has been applied in many different areas. Some studies using *k*-NN for image recognition problems can be listed as follows. Mazzola, S. (1990) used the *k*-NN algorithm in the problem of restoration of damaged digital images. In order to determine the performance of the *k*-NN algorithm in image restoration, real and simulated data have been studied. In the study of Lu,

Y. J., Hsu, Y. H. and Maldague, X. (1992), a modified *k*-NN algorithm was used to classify vehicles on roads. In Vinitski, S. et al., the main purpose is to propose a fast and high accuracy method using the data obtained from 4D feature map. For this purpose, 4 different features were used in the classification of the tissues and the modified *k*-NN algorithm was used. Heikkonen, J. et al. (1997) have studied the problem of classification of urban areas by processing images obtained from satellites. The proposed method consists of feature extraction, feature coding, feature selection and classification steps and the *k*-NN algorithm is used in the classification step, which is the last step. In Jing, J. et al. (1998), a new fuzzy k-NN method is proposed to detect the number of aircraft in the radar from incoherent radar signals. Chien, J. T. and Wu, C. (2002) handled the face recognition problem and used *k*-NN in the classification section of the model they proposed. J. Tian (2010) proposed a solution model for hyperspectral image classification problem by combining *k*-NN and local manifold learning method.

Another important application area where *k*-NN algorithm is used is missing data completion and data discovery problems. In Todeschini, R. (1990), the problem of estimating missing data in multidimensional data is considered. A method based on the *k*-NN algorithm has been proposed for estimating missing data. Batista, G. E. and Monard, M. (2003) aimed to complete the missing data in the data set using *k*-NN in order to obtain more successful results in data analysis. In this work on text mining, Tan, S. and Zhang, J. (2008) used *k*-NN algorithm for segmentation of Chinese documents.

Besides these, *k*-NN is used in hybrid models with different algorithms. In Jain, A. K. and Mao, J. (1991), Artificial Neural Network structure and *k*-NN algorithm have been applied together. The neuron connections in Artificial Neural Network are designed using the *k*-NN algorithm. When the obtained model was examined, it was seen that the classification results were much faster than the standard k-nn, although the results were the same. Buturović, L. J.

(1993) used the *k*-NN algorithm to calculate Bayes error's density function and probabilities. The results showed that the proposed model has improved in bayes errors.

K value is an important parameter that affects classification accuracy in the *k*-NN algorithm. There are many methods in the literature concerning the selection of *k* value, but the best *k* value can be defined by testing different *k* values (Hmeidi, I. et al., 2008).

In *k*-NN applications, data set is divided into 2 as training set and test set. Different cross validation applications as described previously can be performed for this process. The success of the applied model is the ratio of the correctly classified data in test set to the total number of data in test set after classification of the data in the test set using the data in the training set.

Table 3: Synthetic Data Set

	Feature 1	Feature 2	Class
$\mathbf{x_1}$	1.8	3.1	1
$\mathbf{x_2}$	1.6	2.8	1
$\mathbf{x_3}$	1.3	3.8	1
$\mathbf{x_4}$	3.3	4.1	2
$\mathbf{x_5}$	3.9	4.7	2
$\mathbf{x_6}$	4.0	4.6	2
$\mathbf{x_7}$	3.1	1.5	3
$\mathbf{x_8}$	2.8	1.2	3
$\mathbf{x_9}$	3.3	1.3	3

As in the NN algorithm, an example makes the *k*-NN algorithm more understandable. It can be examined at the same time how to change the classification for different *k* values with this example. In Table 3, a synthetically produced data set is presented. This data set has 9 data and 3 classes. A class and 2 features information of each data are given.

The scatter of these samples in the 2D plane is shown in Figure 6. New data Y which has feature 1 and feature 2 values 3 and 3 respectively are tried to classify by using the data given in Table 3 as a training set.

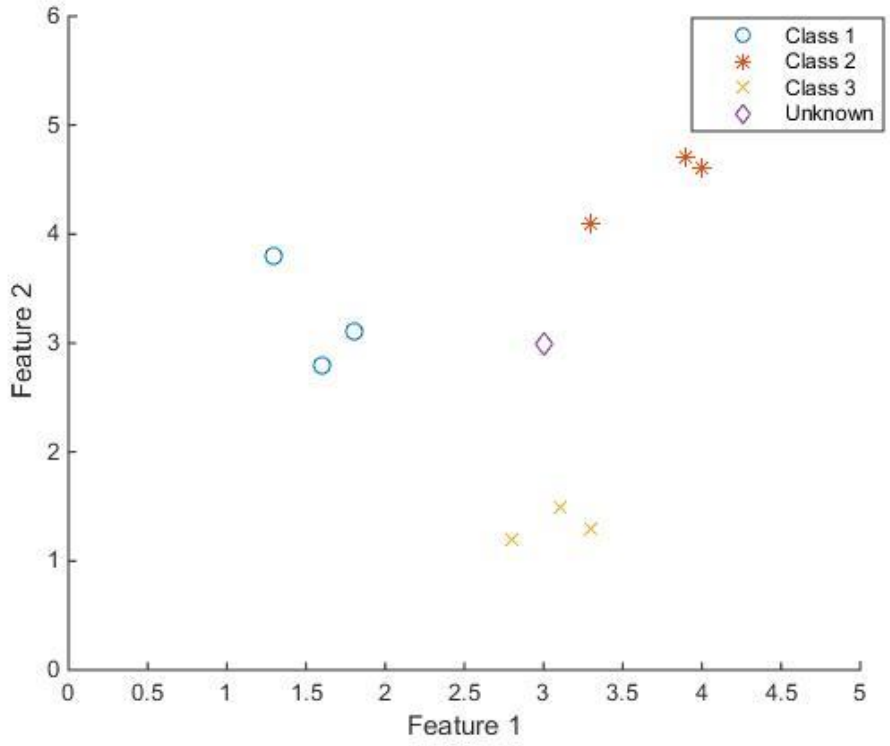

Figure 6: Toy Example: *k*-NN

First, the distances of the Y data and all the data in the training set are calculated using the Euclidean distance measure function and these values are given in Table 4.

Table 4: Distance between Y and Training Data

	Distance	Rank
$\mathbf{x_1}$	1.20	2
$\mathbf{x_2}$	1.41	3
$\mathbf{x_3}$	1.88	7
$\mathbf{x_4}$	1.14	1
$\mathbf{x_5}$	1.92	9
$\mathbf{x_6}$	1.89	8
$\mathbf{x_7}$	1.50	4
$\mathbf{x_8}$	1.81	6
$\mathbf{x_9}$	1.73	5

Secondly, it is checked class to which the Y point is assigned according to the selected *k* value. The most important issue to note here are the classes assigned to Y for different *k* values. In this example, the classes to which the Y data is assigned for *k* = 1, 3, and 6 neighbor values are checked. In Table 5, the nearest *k* neighbor data to Y data for different k values are shown in terms of closeness.

Table 5: *k*-Nearest Neighbors of Y

k value	*k*-nearest neighbor
k=1	$\mathbf{x_4}$
k=3	$\mathbf{x_4, x_1, x_2}$
k=6	$\mathbf{x_4, x_1, x_2, x_7, x_9, x_8}$

For *k* = 1, the nearest neighbor is $\mathbf{x_4}$**.** In this case, Y is assigned to class 2 which is class of $\mathbf{x_4}$**.** For *k*=3, the nearest 3 neighbors are $\mathbf{x_4}$, $\mathbf{x_1}$ and $\mathbf{x_2}$, respectively. $\mathbf{x_4}$ is a member of class 2, however, $\mathbf{x_1}$ and $\mathbf{x_2}$ are members of class 1. Therefore, for *k*=3, Y is assigned to class 1, since the majority of neighbors are in class 1.

Finally, for *k*=6, majority of neighbors, $\mathbf{x_7}, \mathbf{x_9}$ and $\mathbf{x_8}$, belong to the class 3 and Y is assigned to class 3.

As is clear from the example, the choice of *k* for the *k*-NN algorithm has a vital importance to make the correct classification. The different *k* values selected may cause the data in the test set to be assigned to different classes.

2.2.3. Weighted *k*-Nearest Neighbor Algorithm

The frequent use of the *k*-NN algorithm in classification problems has also led to a lot of study being done to improve its performance. The search for the optimum *k* neighbor value and the use of different distance measures are the most common methods to increase the classification accuracy (Jiang, L et al, 2007).

In the literature, the weighted *k*-NN algorithm is used for two different problem definitions. Firstly, the study by Dudani, S. A. (1976), the *k*-nearest neighbors are weighted, unlike *k*-NN, according to their distance to the test data. This study is also called Modified *k*-Nearest Neighbor(M*k*NN). In the second definition of the weighted *k*-NN problem, features are weighted while the distance is calculated. The basic assumption in this approach is that features have a relatively different relevance level to classify data. This approach is called Feature Weighted *k*-Nearest Neighbor(FW*k*NN).

2.2.4. Modified *k*-Nearest Neighbor

In this method, while the class to which the test data is assigned is determined, the *k*-nearest neighbor is weighted according to the distance to test data.

Weights to be assigned to neighbors in this method are shown in Equation (7).

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1} \; if \; d_k \neq d_1 \\ \\ 1 \; if \; d_k = d_1 \end{cases} \quad for \; j = 1 \ldots . k \qquad (7)$$

$$d_k = distance \; of \; k.nearest \; neighbor$$

By using this equation, weights are calculated to be 1 for the nearest neighbor and 0 for the farthest neighbor. Through these weights, unlike *k*-NN, the closeness of neighbors in the M*k*NN algorithm also affect the classification result.

A sample application can provide a clearer understanding of the algorithm. for By using training set and test data which are used in *k*-NN algorithm and given information about in Tables 4 and 5, M*k*NN algorithm is applied for *k*=3. Since *k* = 3, the weights of the closest 3 neighbors for Y test data are calculated.

$$w_1 = 1 \text{ because of } d_k = d_1 \qquad (8)$$

$$w_2 = \frac{1.41 - 1.20}{1.41 - 1.14} = 0.7777 \qquad (9)$$

$$w_3 = \frac{1.41 - 1.41}{1.41 - 1.14} = 0 \qquad (10)$$

The scores of the classes are calculated to determine the class to which test data assigns after weights of neighbors are calculated. The first neighbor belongs to

the class 2 and second and third neighbors belong to class 1. Therefore, the scores of class are calculated as shown in Equation (11) and (12).

$$Score\ of\ class\ 2 = w_1 = 1 \tag{11}$$

$$Score\ of\ class\ 1 = w_2 + w_3 = 0.7777 \tag{12}$$

As it can be seen from the equations, class 2 has a higher score than class 1. Thus, Y test data assigns to class 2. Unlike the *k*-NN algorithm in which Y test data was assigned to class 1, in the M*k*NN algorithm for $k = 3$, Y test data is assigned to class 2.

2.2.5. Feature Weighted *k*-Nearest Neighbor

The standard Euclidean distance function is inadequate to do the correct classification when the data set we use has irrelevant features. The difficulties in correct classification that arise due to the increase in irrelevant features in the data set are called the curse of dimensionality.

To overcome the curse of dimensionality problem, first, feature selection method was proposed. In this method used in a lot of studies, the features with the least relevant are completely extracted from the feature space while the distance between two data is calculated.

After the feature selection approach, the feature weighed method based on the relevance level of features instead of completely removing the features was used to solve the curse of dimensionality problem. In this method, the distances are calculated using the weights of the features.

The FW*k*NN problem is a method used to improve the classification performance of the standard *k*-NN algorithm. The main purpose of FW*k*NN approach is to increase classification accuracy by determining the different relevant levels of features used for classification. To achieve this purpose, different distance measure from the standard Euclidean distance is used. One sample of these functions, called the weighted Euclidean distance function, is given in Equation (13).

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{f=1}^{m} w_f \times \left(x_{if} - x_{jf}\right)^2} \tag{13}$$

In this equation, x_{if} and w_f represent the feature *f* of sample *i* and weight of feature *f*, respectively.

The method of feature weighted *k*-NN with a numerical example can be explained more clearly. The data set in Table 4 can be used for this numerical example. This data set will be accepted as training set and Y data will be used again as test data. One of the most important parameters in feature weighted *k*-NN is the weight set *w* along with *k* neighbor value. The *k* neighbor value in this example will be assumed to be 1 and the distance values will be calculated for 2 different weight sets **w** by using weighted Euclidean distance function given in Equation (13). As the previous examples, data normalization will not be done in this application because features have almost the same interval.

The feature weight set to be used for the first sample is given in Equation (14).

$$\boldsymbol{w} = [0.9\ 0.05] \tag{14}$$

The distance between $\mathbf{x_1}$ and $\mathbf{Y}$ is calculated as shown in Equation (15) by using the data set given in Table 4, Y test data, the feature weighted Euclidean distance function given in Equation (13) and the feature weights given in Equation (14).

$$d(\mathbf{x_1}, \mathbf{Y}) = \sqrt{0.9 \times (1.8 - 3)^2 + 0.05 \times (3.1 - 3)^2} \cong 1.14 \tag{15}$$

It is clear that the calculated distance between $\mathbf{x_1}$ and $\mathbf{Y}$ in Table 5 is different from the distance calculated in Equation (15). The reason of this difference is that the distances in Table 4 are calculated using the standard Euclidean distance function, however, in this example, the distances are calculated using the weighted Euclidean distance function.

Table 6: Calculated Distances with weight set in Eq.14

	Distance	Rank
$\mathbf{x_1}$	1.14	7
$\mathbf{x_2}$	1.33	8
$\mathbf{x_3}$	1.62	9
$\mathbf{x_4}$	0.38	2
$\mathbf{x_5}$	0.93	5
$\mathbf{x_6}$	1.01	6
$\mathbf{x_7}$	0.35	1
$\mathbf{x_8}$	0.44	3
$\mathbf{x_9}$	0.47	4

The distances between all the data in training set and Y and rank are given in Table 6. For the k = 1 neighbor value, the Y test data is assigned to the class of the nearest neighbor $\mathbf{x_7}$, class 3. However, in the application using standard Euclidean distance, the y test data was assigned to the class of $\mathbf{x_4}$, nearest neighbor of **Y**, class 2.

To see more clearly how different weights affect the classification result, the distances are calculated by using the feature weight set given in Equation (16).

$$\mathbf{w} = [0.1\ 0.8] \tag{16}$$

Calculated distances and rank are presented in Table 7.

Table 7: Calculated Distances with weight set in Eq. 16

	Distance	Rank
$\mathbf{x_1}$	0.39	1
$\mathbf{x_2}$	0.48	2
$\mathbf{x_3}$	0.89	3
$\mathbf{x_4}$	0.99	4
$\mathbf{x_5}$	1.55	8
$\mathbf{x_6}$	1.47	6
$\mathbf{x_7}$	1.34	5
$\mathbf{x_8}$	1.61	9
$\mathbf{x_9}$	1.52	7

The closest neighbor to the **Y** test data is $\mathbf{x_7}$ according to the calculation using weight set in Equation (14). In this case, **Y** is assigned to class 1, unlike the above example.

CHAPTER 3

FOREST OPTIMIZATION ALGORITHM

Forest optimization algorithm (FOA) was proposed by Ghaemi and Feizi-Derakhshi (2014). As it can be understood from the name, FOA is an algorithm that simulates the life processes of the trees in the forests. The FOA algorithm imitates the seed stage, the growth stage, production of new trees stage and the death stage of trees. In the seed stage, some of the seeds that are produced fall into the vicinity of the parent tree, however, some of the seed is thrown away by different means such as winds, animals, and insects, etc. The seeds that find the adequate condition to survive in the place where they fall grow and become trees. Surviving trees produce new seeds under appropriate conditions and die ultimately. FOA is an algorithm that mimics all the processes of trees life.

FOA is a solution method that is used directly to solve different problems, Ghaemi and Feizi-Derakhshi (2016), Maadi, M et al. (2016), and Chaghari, A. (2016), and also frequently compared with other algorithms in terms of performance.

3.1. Life Cycle Of Trees

As in all living things, there are three basic parts in the lives of trees; to be born, to grow and to die. Falling to the ground as a seed is the first step in the lives of the trees. Despite the fact that the trees produce a large number of seeds for the continuation of generations, only a small proportion of these seeds have a chance of sprouting. The main purpose of the trees to produce a large number of seeds

is to ensure the continuity of the generation. This method aims to increase the chances of sprouting from the seeds of the trees. At this stage, the seeds falling to the soil are often close to the parental tree, from time to time, the seeds may fall away from their parents through different factors such as animals and winds. Spreading the seeds to different places increases their ability to continue their lives. The seeds that fall near their parents can only survive in regions with the most optimal living conditions in that region. Finding a large number of trees and seeds in the area makes competitions conditions difficulties and reduces the possibility of survival. We can easily say that after living in a few generations, it is almost impossible for new generations to find life in that region. For all these reasons, the seeds falling far enough away from their parents have a very vital significance for the continuation of the generations of trees. Even though the number of seeds falling away from their parents is fewer than the others and the probability of survival of seeds falling near their parents is much higher than falling away, the task of to find the place having the best living conditions for future generations belongs to them.

The second part of the life of trees is the period of growth. In this period, those who are lucky in the seeds falling to the earth, continue their lives. These young trees have a fundamental importance for the continuity of the forest. These individuals, who were first sprouting and then adult trees, undertake the task of producing new seeds which are the most important part of this phase.

The final stage is the death phase. At this stage the trees lose their life functions, primarily to produce seeds, as a result life cycles end and they leave the forest.

3.2. The Forest Optimization Algorithm's Inspiration From The Life Cycle of Trees

Evolutionary algorithms have developed an inspiration from the processes of living things in nature. FOA is also inspired by the growth of the forest. The fact that forests have been able to survive for thousands of years even under difficult living conditions is the greatest indication of how well they have achieved the optimum conditions for the survival of their lives. Therefore it is a very reasonable approach to develop an evolutionary algorithm by simulating the forms of existence of trees.

In Ghaemi and Feizi-Derakhshi (2014)'s study, they developed FOA as an evolutionary algorithm by observing how forests protect their lives through trees and by shaping the information they have acquired.

FOA contains three basic parts, (i) **local seeding** of the trees, (ii) **population limitation** and (iii) **global seeding** (Ghaemi and Feizi-Derakhshi,2014). The purpose of these steps is to reflect the life cycles of the trees into the algorithm and to solve the problems we face in real life by imitating their survival success.

In FOA, first of all, the initial solutions, called trees in this algorithm, solution space is created as in other evolutionary algorithms. In FOA, each tree has an age which is checked through the iterations. Each tree is a vector which represents the dimensions of the solution to the problem and the age of the tree. The *age* of a tree is set to zero, initially. After the *initial solution space* is created, in **local seeding stage,** new young trees with *zero age* are created with the *local seeding operator* representing the seeds falling close to their parents in the real tree life cycle. At this stage, the ages of the trees in the population except the young ones which is newly produced are increased by one. The fitness function values of all trees in the population are calculated and the trees are sorted according to their fitness function values. In the next stage, **population**

limitation, the population is reduced first according to the *age limit* and then to *the population limit parameter*. Firstly the trees exceeding the *age limit* are removed from the population, then the number of trees in the population is controlled according to *the population limit*. If there are the trees which exceed *the age limit* or the number of trees exceeds *the population limit*, these trees are sent to *the candidate population*. Flowchart of FOA is given in Figure 7.

The **global seeding stage** represents that seeds, falling farther from their parents, which have vital significance in the real life cycle of the trees. This step encourages and ensures that the algorithm will get rid of local solutions and search for global solutions. In the last stage, **population limitation**, the population obtained is sorted again according to the calculated fitness function values and the *age* of the tree with the highest value is changed to zero. The population is checked again according to *age* and *population limit* conditions and trees which do not meet the requirements are sent to the candidate population in where trees removed from the forest and used in the global seeding phase are hold. Then, the algorithm returns to the local seeding stage and these steps are repeated until the stopping conditions are met.

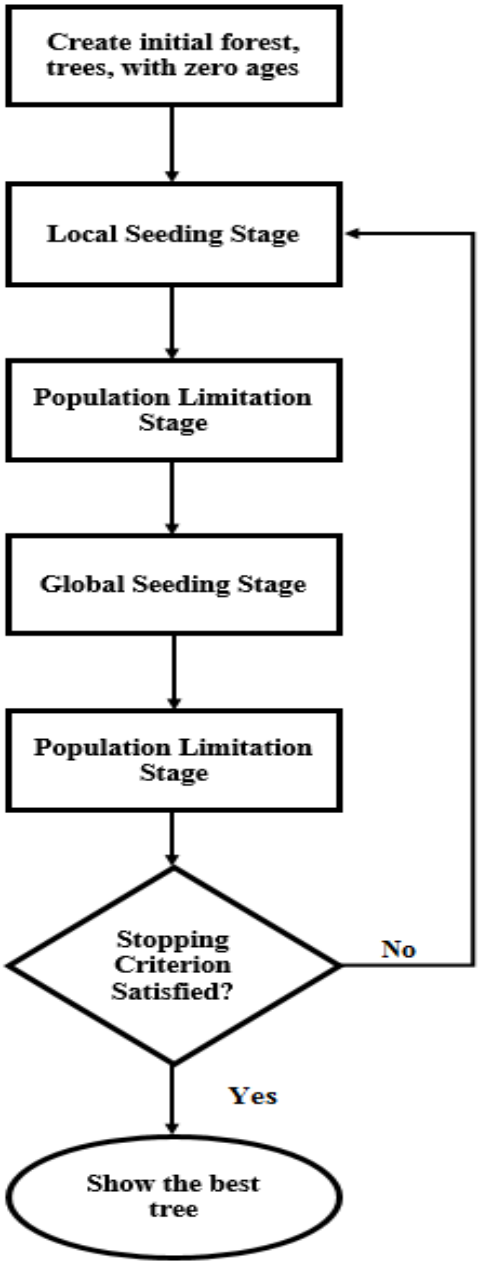

Figure 7: Flowchart of Forest Optimization Algorithm

3.3. Parameters of the Forest Optimization Algorithm

In FOA, there are some parameters that must be set to be used in the steps of the algorithm. These parameters differ for each problem in concern and these

parameters directly affect the success of the algorithm. The first parameter is the age limit parameter called *"Life Time"* used to remove elderly candidates from the population. The other parameter is *"Local Seeding Changes(LSC)"* which determines the number of new trees that the parent tree will generate during **local seeding**. *"Area Limit"* is used to limit the population size in **population limitation** stage and it indicates the maximum number of trees in the population. **In global seeding** phase, two different parameters are used, *"Transfer Rate"* and *"Global Seeding Changes(GSC)"* parameters. *"Transfer Rate"* parameter indicates the percentage of the trees in the candidate population will be added to the solution space. For example, if the transfer rate is 0.1 and the number of trees in the candidate population is 100, the number of trees entering the solution space in the global seeding phase is set to 10. *GSC*, another parameter in global seeding, shows number of elements(decision variables) of the chosen tree will change in the solution space. The last parameter is the *stopping condition* parameter. This parameter can be defined as a certain number of iteration, a special accuracy level or unchanging best solutions for several iterations (Ghaemi and Feizi-Derakhshi,2014).

3.4. Steps of the Forest Optimization Algorithm

The FOA basically consists of six steps. These steps can be listed as Creating Initial Trees, Local Seeding, Population Limitation, Global Seeding, Ranking the Trees and Stopping Condition.

3.4.1. Creating Initial Trees

In this algorithm, each tree is a vector of solutions for the specified problem. In addition to the potential solution values, the *age parameter*, indicating a number

of times the solution was in the solution space, is also stored in the tree. If the size of the solution space of the problem is considered by n, each tree will be,

$$Tree : [x_1, x_2, x_3, x_4, \dots, x_n, Age]$$

In “Creating Initial Trees” stage, trees with zero age are created until the *population limit* is reached.

3.4.2. Local Seeding

This stage simulates the seeds falling near their parents. At this stage, new trees that are similar but not identical to parent trees which have zero age are created. The number of new trees to be generated from the parent trees is determined by *LSC parameter*. For example, in the case LSC is 5, a parent with age 0 allows 5 more trees to participate in the population as a result of local seeding. After this operation, the age of the parent tree is increased by one and new trees are generated at the age of zero. For a clear understanding, this example is shown in Figure 8.

New trees formed during local seeding are the result of changing only one variable of the parent tree. The variable to be changed is randomly selected and a random number that will not cause the selected variable to exceed the limit range is added to this selected variable.

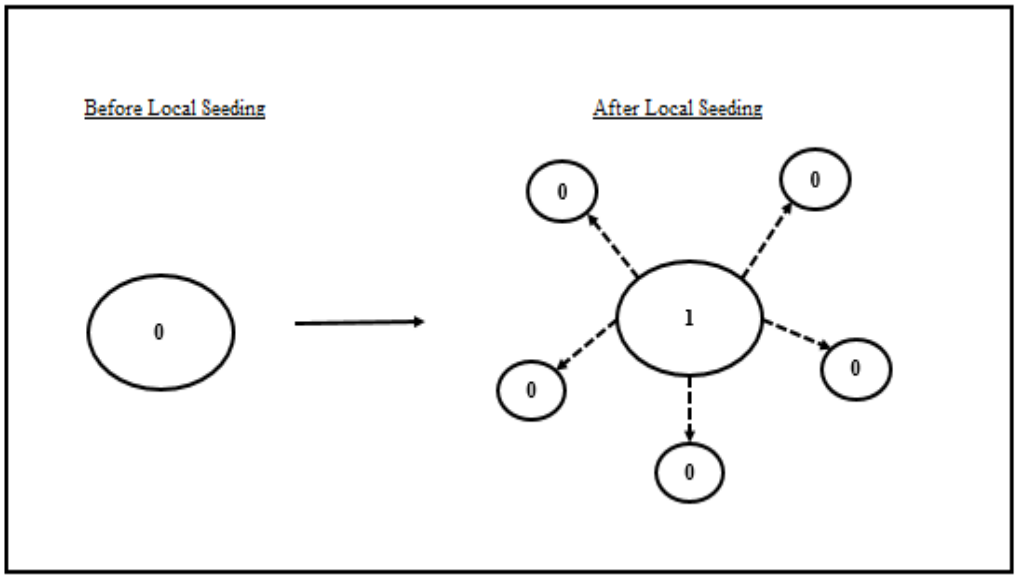

Figure 8: Local Seeding, LSC=5

For example, a tree having six variables (see Figure 9), in a situation that LSC is 2, the first variable to be changed will be the first component and the second will be the third component. For the first component, the generated random number is -0.2 and the random number generated for the second component is 0.7.

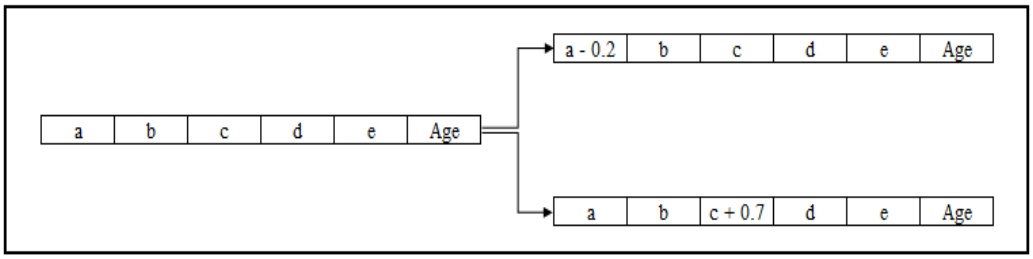

Figure 9: Local Seeding, *LSC*=2 and Random Numbers : [-0.2 0.7]

3.4.3. Population Limitation

Population limitation is controlled in two different ways using two parameters, "*Life Time*" and "*Area Limit*". Trees in the solution space exceeding the "*Life Time*" parameter in the forest are eliminated from the forest and sent to the candidate population. Trees are sorted based on their fitness function values in the solution space, the ones, exceeding the "*Area Limit*" parameter in the forest, are eliminated from the forest and again they are sent to the candidate population.

3.4.4. Global Seeding

Global seeding is a step that imitates the seeds that fall away from the parent tree in the life cycle of the trees. The seeds produced by the parent trees can be transported away from the parent tree through intermediaries such as birds, animals, water, wind. This process allows the trees to search a better place for their future in the forest.

In the global seeding phase, two previously defined parameters are used. These parameters are "*Transfer Rate*" and "*Global Seeding Changes*". First of all, the trees at the transfer rate from the candidate population are selected to be included in the solution space. Secondly, according to *GSC parameter,* the variables in the selected trees are determined randomly and this variable is replaced with a randomly generated value. It is important that generated random value must be in the solution interval of the selected variable. In the global seeding phase, briefly, the *Transfer Rate parameter* indicates the number of trees to be included in the forest, and *GSC* indicates the number of variables that will change in the selected trees. For instance, assume that the number of trees in the candidate population is 10, *Transfer Rate parameter* is 0.1, and *GSC* is 2. Randomly selected variables are 1 and 3 for the first child, 2 and 4 for the second child. The solution range of the selected variables is defined as [-3 3]. The four random

numbers generated in the given range are 1.7, -2.4, -0.3 and 0.9, respectively. Figure 10 shows the global seeding process under the above conditions. As a result of this process, two new trees with zero age were added to the forest.

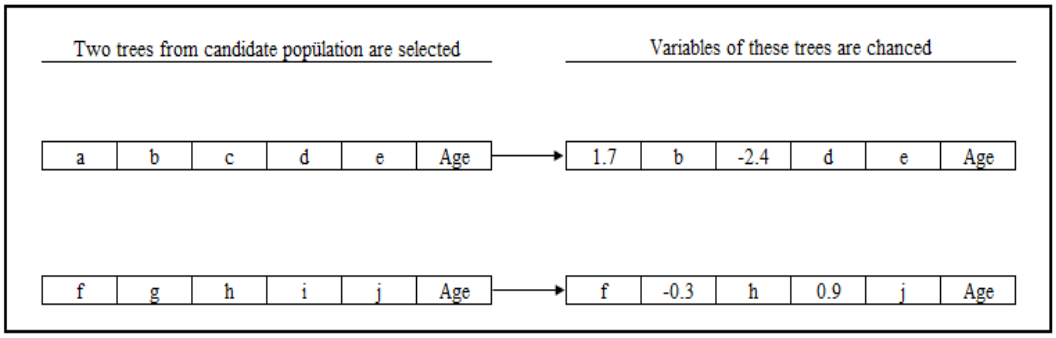

Figure 10: Global Seeding Example with *GSC*=2

3.4.5. Ranking the Trees

The ranking of the trees according to the fitness function values must be made after each stage of the new trees added to the forest, i.e., after the local seeding and global seeding steps. Also, the age of the best tree in the forest is updated to zero after the Global Seeding phase.

3.4.6. Stopping Condition

Condition states the criterion that the algorithm is terminated. As in other evolutionary algorithms, this criterion can be defined as a certain number of iteration, a special accuracy level or unchanging best solutions for several iterations.

As a result, all stages of the FOA and the parameters used in these stages are explained in detail in this section. The pseudo-code of the algorithm is given in Table 8.

Table 8: Pseudo-Code of Forest Optimization Algorithm

Pseudo-Code of Forest Optimization Algorithm

Step 0: *Define parameters*; Area Limit, Life Time, LSC, GSC, Transfer Rate
Step 1: *Initialization:* Create the initial forest with random trees with zero age,
Step 2: *Iteration:* While stop condition is not satisfied do,
Step 2.1: *Local seeding* to trees with zero age
for i=1:LSC,
Select a variable randomly from the selected tree,
Add a small random value to the selected variable,
end
Increase by 1 the age of all other trees except the new ones,
Step 2.2: Calculate fitness value of all trees and sort them according to fitness value,
Step 2.3: *Population Limiting*
Remove the trees that have bigger age than the life time parameter and send them to the candidate population,
Remove the trees exceeding the area limit from the end of forest and send them to the candidate population,
Step 2.4: *Global Seeding*
Choose trees from candidate population according to transfer rate parameter,
for i=1:number of selected trees,
Select randomly "GSC" variables of selected tree,
Change the selected variables with randomly generated value which must be in the selected variable range,
Make the age of selected tree zero and add the tree to the forest,
end
Step 2.5: Calculate fitness value of all trees and sort them according to fitness value,
Step 2.6: Make the age of best tree zero
Step 2.7: *Population Limiting*
Remove the trees that have bigger age than the life time parameter and send them to the candidate population,
Remove the trees exceeding the area limit from the end of forest and send them to the candidate population,
Step 3: *Results:* Show the best tree as a result

3.5. Testing Forest Optimization Algorithm

As a common practice in the literature, performance of an evolutionary algorithm is tested with some test functions existing in the literature. These test functions can be classified according to different properties. These properties used during classification represent the difficulty of finding the solution of the test function. Test functions can be classified as unimodal or multimodal, two-dimensional or multidimensional; or based on the number of extreme-points such as a small number of local extremes or the huge number of local extremes (Molga and Smutnicki, 2005).

FOA was tested with some nonlinear test functions to see its performance in nonlinear problems. These functions were selected from Ghaemi and Feizi-Derakhshi (2014) and Molga and Smutnicki (2005). All functions have a continuous solution space. In order to be able to present the graphs of solution spaces, these functions have been discussed with two variables. These functions and related information are given in Table 9 and the 3D graphics of these functions are presented in Appendix A.

FOA algorithm for each of these functions was initiated 30 times. Algorithm parameters for each function are given in Table 10; *Area Limit*: 30, *Life Time*:6, *LCS*:1, *GSC*:1, *Transfer Rate*:0.1. While the initial solution space(forest) was created, trees were randomly selected at the beginning of the algorithm according to the interval of related variables of the function. The new trees created during the **local seeding stage** are differentiated from the parent trees with small values generated randomly. This stage serves as local search in solution space. New trees created during the **global seeding stage** show large differences from the parent tree as they remain within the solution interval of the variables. This process work as the global search in the solution space.

Table 9: Test Functions

Func.	Name of Function	Equation	Ranges of Variables
F1	-	$f(x_i) = x_i \times \sin(4x_i) + 1.1 \times x_2 \times \sin(2x_2)$	$0 < x_i < 10$
F2	De Jong's Function	$f(x_i) = \sum_{i=1}^{n} x_i^2$	$-5.12 < x_i < 5.12$
F3	Axis Parallel Hyper-Elipsoid Function	$f(x_i) = \sum_{i=1}^{n} i \times x_i^2$	$-5.12 < x_i < 5.12$
F4	Rosenbrock's Valley Function	$f(x_i) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$-2.048 < x_i < 2.048$
F5	Rastrigin's Function	$f(x_i) = 10n + \sum_{i=1}^{n} [x_i^2 - 10 \times \cos(2\pi x_i)]$	$-5.12 < x_i < 5.12$
F6	Schwefel's Function	$f(x_i) = \sum_{i=1}^{n} \left[-x_i^2 \times \sin\sqrt{\lvert x_i \rvert}\right]$	$-500 < x_i < 500$
F7	Griewangk's Function	$f(x_i) = \left({}^{1}/_{4000}\right) \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{i}\right) + 1$	$-600 < x_i < 600$
F8	Sum of Different Power Function	$f(x_i) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$-1 < x_i < 1$
F9	Michalewicz's Function	$f(x_i) = -\sum_{i=1}^{n} \sin(x_i) \left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2\mathrm{m}}$	$0 < x_i < \pi$
F10	Branin's Function	$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 + d)^2 + e(1 - f)\cos(x_1) + e$	$-5 < x_1 < 10$ $0 < x_2 < 15$
F11	Easom's Function	$f(x_1, x_2) = -cos(x_1)\, cos(x_2) exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$-100 < x_i < 100$
F12	Goldstein's Function	$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	$-2 < x_i < 2$
F13	Six-Hump Camel Back Function	$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + (-4 + 4x_2^2) x_2^2$	$-3 < x_1 < 3$ $-2 < x_2 < 2$
F14	"Drop Wave" Function	$f(x_1, x_2) = -\left(1 + \cos(12\sqrt{x_1^2 + x_2^2})\right) \Big/ \left(\frac{1}{2}(x_1^2 + x_2^2) + 2\right)$	$-5.12 < x_i < 5.12$
F15	Shubert's Function	$f(x_1, x_2) = \sum_{i=1}^{5} i\, cos\big((i+1)x_1 + i\big) \times \sum_{i=1}^{5} i\, cos\big((i+1)x_2 + i\big)$	$-5.12 < x_i < 5.12$

3.5.1. Some Examples From Test Function

Selected test functions are multimodal, nonlinear and they have many local extreme points. Because of these features, these functions have often been used in the course of testing many evolutionary algorithms.

In this section, the graphs of some of the test functions specified in Table 9 and values found by FOA out of 30 initializations will be presented. In addition, there will be a graphical representation of fitness values based on the iteration for one FOA initialization selected for each function. The FOA parameters used in this study are given Table 10.

Table 10: FOA Parameters for Test Functions

Area Limit	Life Time	Transfer Rate	LSC	GSC
30	6	0.1	1	1

First test function F1is given in Equation (17).

$$f(x,y) = x \times \sin(4x) + 1.1 \times y \times \sin(2y)\,,\ 0 < x,y < 10 \qquad (17)$$

The graph of F1 for two variables is shown in Figure 11. As it can be seen clearly from the graph, F1 function has nonlinear structure with many local solution points.

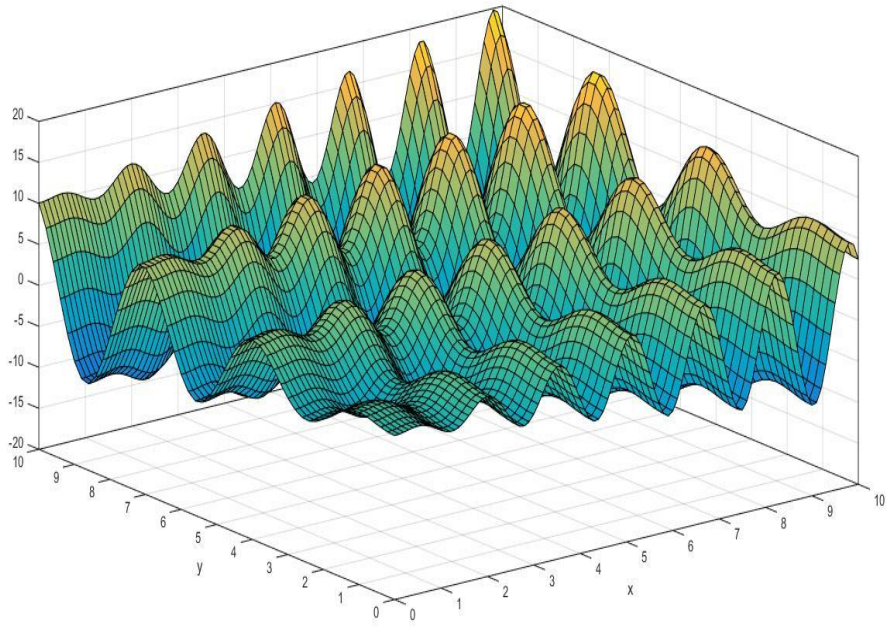

Figure 11: 3D Plot of F1 Function

The FOA has been implemented with 30 initialization for this function and the fitness values obtained from each initialization are presented in Figure 12. The best fitness function value for F1 function is -18.5547. When the results obtained using the FOA in Figure 12 are examined, it is seen that the FOA has achieved the best result 5 times out of 30. Also, the worst result of the FOA in 30 initializations is -18.5539 and this value is very close to the best value (9E-6%).

Iteration-based fitness value of best and worst trees and population average fitness value for a random initialization are shown in Figure 13. In this graph, for the F1 function, it seems how FOA converges to the best result.

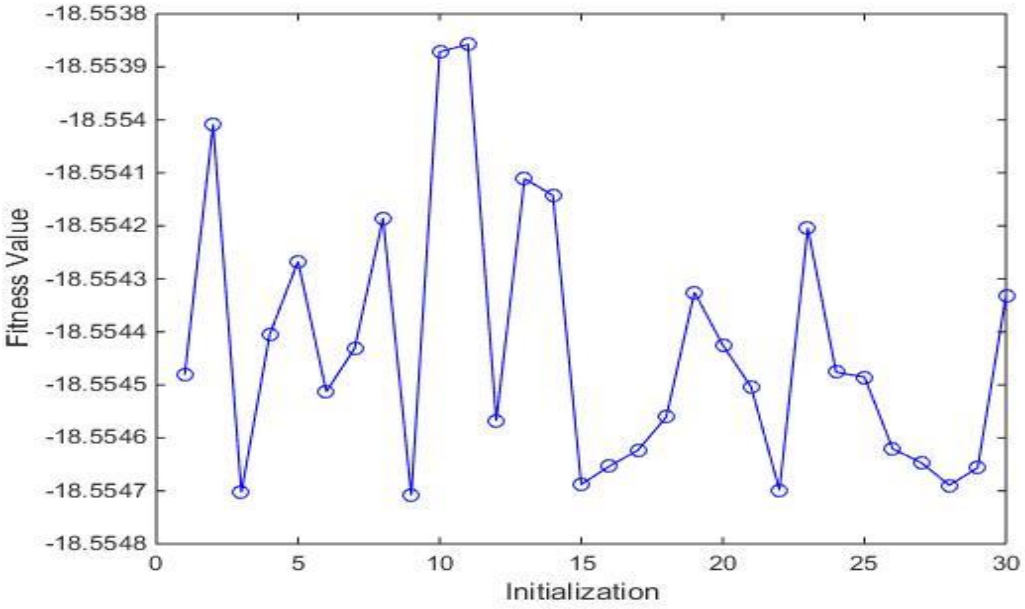

Figure 12: Results of FOA for F1 Function

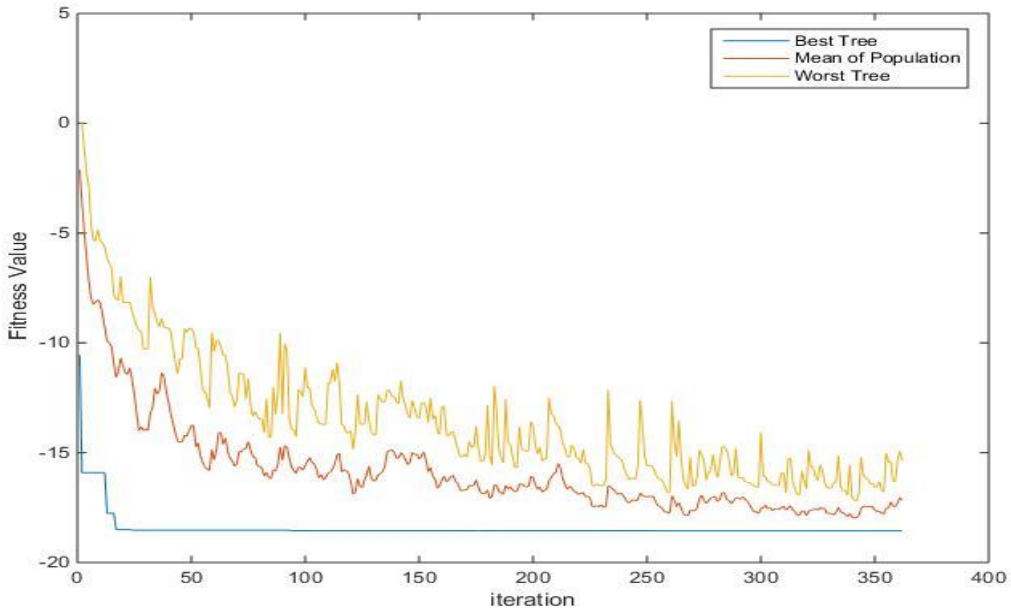

Figure 13: Fitness Value of Best and Worst Tree in Population and Mean of Population for F1

The second function considered is F6 represented in Equation (18).

$$f(x_i) = \sum_{i=1}^{n} \left[-x_i^2 \times \sin\sqrt{|x_i|} \right], \quad -500 < x_i < 500 \tag{18}$$

The graph of F6 for two variables is shown in Figure 14. The function F6 is also a difficult nonlinear function with a large number of local extreme points.

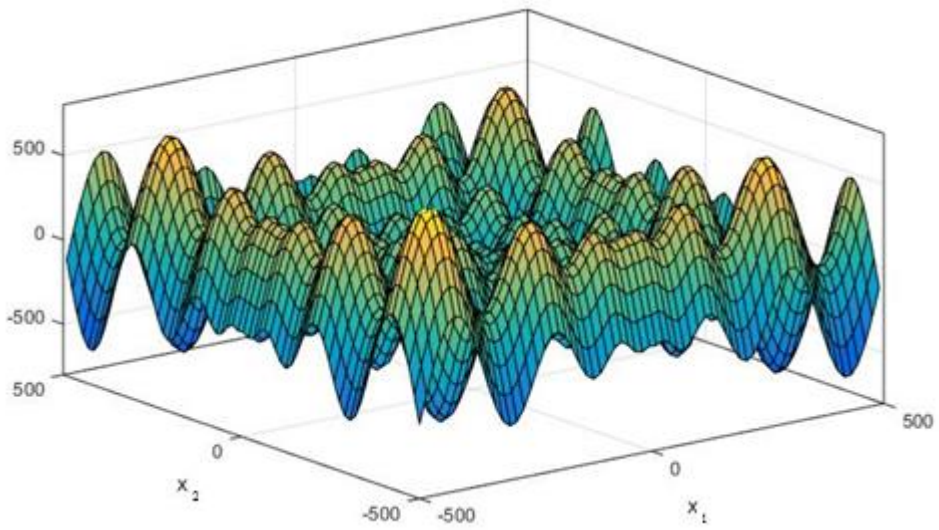

Figure 14: 3D Plot of F6 Function

In Figure 15, the results of 30 initializations obtained by applying FOA are given. The best fitness function value for the F6 function is -836.9658. The worst result of FOA for 30 initialization is -836.9657 which is very close to the best fitness value (1E-6%). The FOA achieved the best result for F6 function at 16 out of 30 initializations.

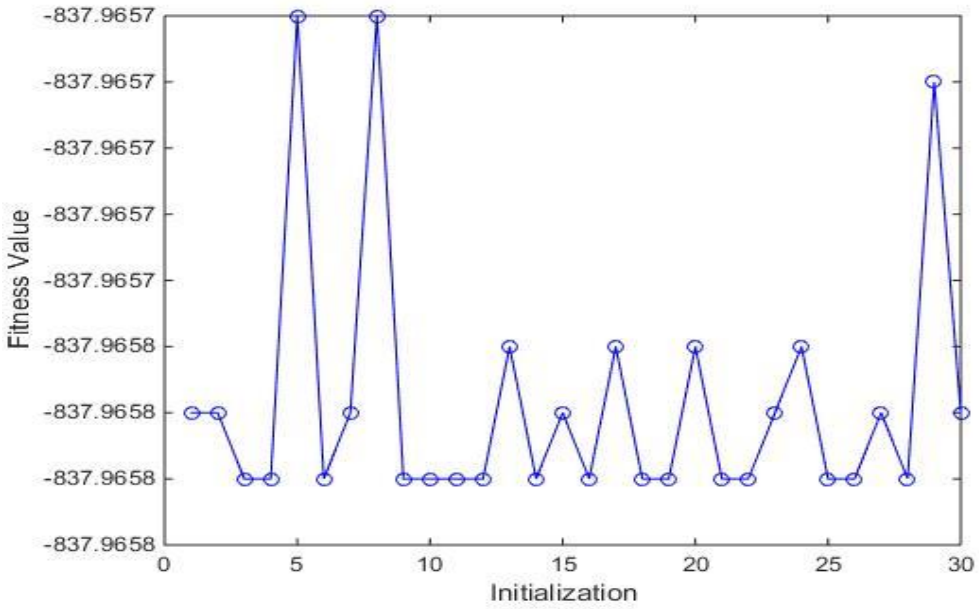

Figure 15: Results of FOA for F6 Function

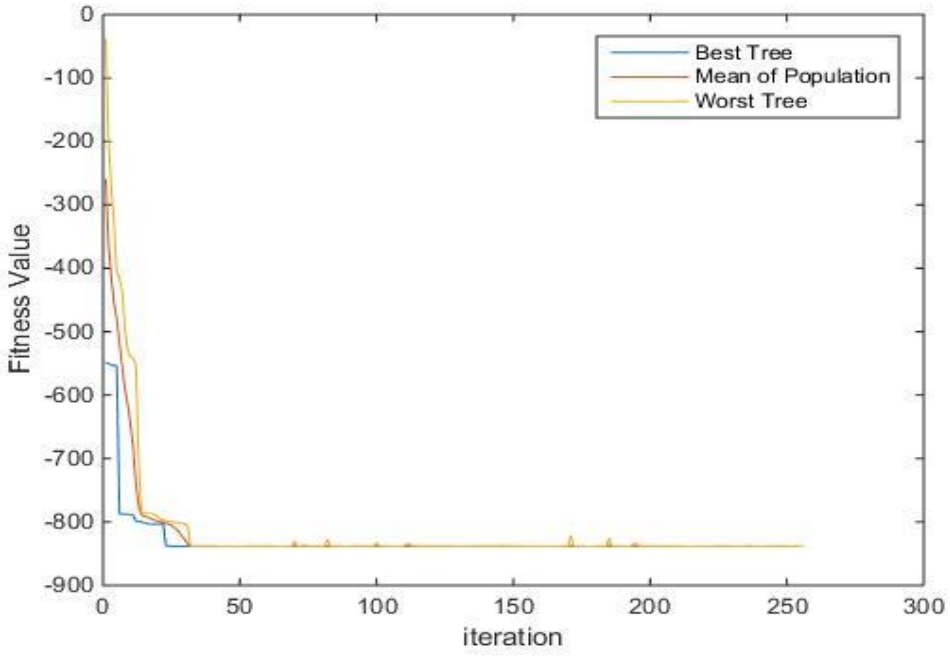

Figure 16: Fitness Value of Best and Worst Tree in Population and Mean of Population for F6

It appears that the FOA has converged to the best fitness value very quickly in F6, in the examination of Figure 16. The population also evolves quickly to the best solution.

It can be easily understood from Table 11 and the two detailed examples F1 and F2, FOA have achieved successful results for the test functions.

3.5.2. All Results in Test Functions

In this part of the study, values such as the best fitness value of each initialization, the state of populations in each iteration, the working time of the algorithm are recorded. The best, worst, average fitness value for each function, the best fitness value in the literature, the approximate value of the best fitness value to the best value in the literature, the best, worst and average working times of FOA are presented in Table 11.

When Table 11 is analyzed, it is clear that the best results achieved by the FOA are almost identical to the best results reported in the literature. FOA has achieved the worst accuracy result with 2.95E-7 % for F15. To be more clear, the best value in the literature is -186.7309 while the worst value obtained by the FOA is -186.5488. When the FOA's solution times are analyzed, the worst was achieved for F10 with 40.8190 seconds. The best time for the same function is 2.0150. When total times for 30 initializations are examined, the total time for almost all functions is less than 300 seconds. This means that the average time for each iteration is less than 10 seconds. It is clear that the FOA has converged to the best result in a short time if the performance/time evaluation is considered.

Table 11: Results of FOA

Func.	Best of FOA	Worst of FOA	Average	Standard Deviation	Best Value in Literature	Percentage Accuracy	Best Time	Worst Time	Total Time
F1	-18.5547	-18.5539	-18.5544	0.0002	-18.5547	0	2.0440	37.6080	350.0670
F2	0.0000	0.0000	0.00001	1.25E-05	0.0000	1,45E-11	1.1960	39.3290	260.3380
F3	0.0000	0.0001	0.00001	2.12E-05	0.0000	7,16E-10	2.0470	26.8770	295.3580
F4	0.0000	0.0010	0.00024	0.0003	0.0000	1,06E-08	1.7640	31.3720	258.7280
F5	0.0000	0.0012	0.0004	0.0003	0.0000	7,64E-08	2.2310	21.6570	338.3160
F6	-837.9657	-837.9657	-837.966	1.92E-05	$-418.9829 \times n$	2,55E-07	3.0760	35.9960	230.7980
F7	0.0000	0.0117	0.00392	0.00420	0.0000	6,9E-09	1.9440	31.1310	357.0560
F8	0.0000	0.0000	0	3.64E-06	0.0000	5,72E-12	2.0610	39.5670	297.0500
F9	-1.8013	-1.7995	-1.80072	0.0007	$-1.8013 \ for \ n = 2$	0	1.2520	18.5570	203.4660
F10	0.3979	0.3986	0.39804	0.0002	-0.3979	1,99E-08	2.0150	40.8190	223.3170
F11	-1.0000	-0.9988	-0.99985	0.0003	-1.0000	3,56E-09	2.4410	16.8410	215.5120
F12	3.0000	3.0167	3.0044	0.0048	3.0000	1,27E-07	1.3090	35.9510	283.9690
F13	-1.0316	-1.0311	-1.03156	0.0001	-1.0316	0	1.4080	28.0040	255.6170
F14	-0.9999	-0.9907	-0.99857	0.0019	-1.0000	1,38E-07	2.4130	35.9870	319.5480
F15	-186.7308	-186.5488	-186.704	0.0358	-186.7309	2,95E-07	3.8310	24.2970	285.6860

As a result, by taking into consideration of the results of FOA on test functions, it can be obviously said that FOA seems to be a successful solution algorithm to unimodal or multimodal, two-dimensional, and the number of extreme-point such as a small number of local extremes or the huge number of local extremes nonlinear and continuous functions.

CHAPTER 4

FEATURE WEIGHTED *k*-NEAREST NEIGHBOR PROBLEM

The *k*-nearest neighbors (*k*-NN) algorithm is one of the most commonly used algorithms among the classification algorithms. The most fundamental reason its usage is easy implementation and easy adaptation to all kinds of problems.

In the standard *k*-NN algorithm implementation, it is assumed that all features have the same level of relevance for classification. Under this assumption, in standard application of the *k*-NN, the objective is to find the optimal *k* value for the best classification accuracy.

k value is tried to optimize in the standard *k*-NN algorithm, but with feature weighting problem applied to *k*-NN, the problem became both the optimization of features weights **w** and *k* value optimization in weighted *k*-NN applications. Feature weighted *k*-NN in literature, feature weights are sought to increase the classification accuracy for a predetermined *k* value.

In the feature weighted *k*-NN algorithm, the goal is to find the weight set for the features and *k* value to maximize the classification accuracy. As it can be seen here, the feature weighted *k*-NN problem has two parameters; *k* nearest neighbor and **w** feature weight set. The feature weighted *k*-NN classification accuracy function used in this study is defined as follows.

n= # of samples in test set,

$c(\mathbf{x}) =$ class of x,

$c_{(k,\boldsymbol{w})}{}'(\mathbf{x}) =$ represent the class to which x is assigned according to **w** weighted distance and *k* nearest neighbor value,

The feature weighted *k*-NN algorithm can be expressed as a function depending on **w** and *k* as follows;

$$\delta(a,b) = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases} \tag{19}$$

$$\arg max \left[\mathrm{f}(k,\mathbf{w}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \delta\left(\mathrm{c}(\mathbf{x_i}), \mathrm{c}_{(k,\mathbf{w})}{}'(\mathbf{x_i})\right)/\mathrm{n} \right] \qquad (FW)$$

As a mentioned early, in the feature weighted *k*-NN, the accuracy of classification is determined by examining how much of the data in the test set is correctly classified. Accordingly, in problem (FW), the class to which a sample **x** in the test set belongs and the class to which the feature weighted *k*-NN algorithm assigns sample **x** are compared. In this comparison the indicator function defined in Equation (19) is used. The process continues until all the samples in the test set are checked.

In this study, the feature weighted *k*-NN is called problem P1 and is shown in problem (FW). As it is seen in problem (FW), the accuracy of classification depends on the *k* value and the feature weight set, **w**, parameters. As it is understood from this definition of problem P1, actually the feature weighted *k*-NN problem is an optimization problem where two parameters must be

optimized. The first optimization problem is to find the *k* value that will maximize the accuracy of the classification as it is in the standard *k*-NN. The second optimization problem is to find feature weights set, **w**, to maximize the classification accuracy.

Feature weighted *k*-NN problem have nonlinear, multimodal, multidimensional solution space having a huge number of local extreme points because of the weighted Euclidean distance function. Figure 17 shows the classification accuracy values graph for IRIS data set for 74 different weight sets and neighbor values from *k*=1 to *k* = 20.

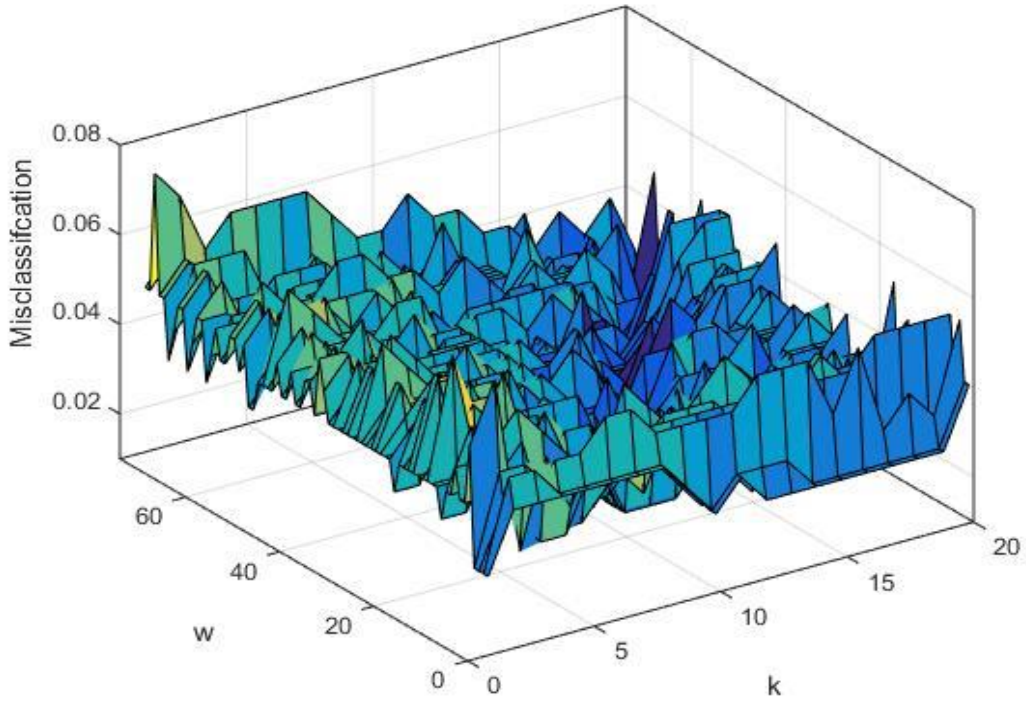

Figure 17: 3D Graph of Classification Accuracy Function

In this study, to solve problem P1 two different algorithms have been proposed. In both of the proposed algorithms, the main goal is to optimize both feature weights and *k* values to maximize classification accuracy.

The first proposed algorithm is called "Algorithm 1", in which the *k* and **w** values are optimized an iterative scheme and iterations are applied until the stop criterion is satisfied.

In the second algorithm called "Algorithm 2", the *k* and **w** values are optimized simultaneously and Forest Optimization Algorithm is used to solve the problem.

Below, two proposed algorithms are described in detail. Flow charts and pseudo-codes have been provided for both algorithms. Finally, each algorithm has been tested via six different data sets and their performances have been reported.

4.1. Algorithm 1

As mentioned early, feature weighted *k*-NN is a problem that both the feature weights **w** and *k* values are optimized. Because of the nonlinear nature of the problem, as well as the many local solution points, to find the best solution for this problem is not trivial.

The classification accuracy function of feature weighted *k*-NN shown in problem (FW) is handled by dividing two subproblems in Algorithm 1. In both subproblems, the main purpose is to maximize the classification accuracy.

$$\arg max \left[\mathrm{f}(k) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \delta \left(\mathrm{c}(\mathbf{x_i}), \mathrm{c}_{(k,\mathbf{w})}{}'(\mathbf{x_i}) \right) / \mathrm{n} \right] \text{ with fixed } \mathbf{w} \qquad (SP1)$$

In the subproblem 1 shown in problem (SP1), **w** is fixed and the *k* value is optimized. The process in subproblem 1 is the same as the standard *k*-NN application. The only difference is that the distances between the samples are calculated using the feature weights.

$$\arg max \left[\mathrm{f}(\mathbf{w}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \delta \left(\mathrm{c}(\mathbf{x_i}), \mathrm{c}_{(k,\mathbf{w})}{}'(\mathbf{x_i}) \right) / \mathrm{n} \right] \text{ with fixed } k \qquad (SP2)$$

In the subproblem 2 shown in problem (SP2), the other parameter *k* is fixed and **w** is optimized. Feature weighting in *k*-NN is performed to maximize the classification accuracy in this subproblem.

In Algorithm 1, the two subproblems described above are solved iteratively. This algorithm alternate between,

(1) Subproblem 1: Neighborhood Problem with given weight set **w,**
(2) Subproblem 2: Weighting Problem with given *k* value,

until the same *k* value is obtained consecutively.

This algorithm is an iterative algorithm and *k* value and **w** are optimized iteratively. First, in the subproblem 1, the optimal *k* value is searched by using the standard *k*-NN algorithm, assuming that all features have the same relevance level, in other words, *k* value is optimized. While the *k* value is optimized, the weighted Euclidean distance function is used where the distances between samples are calculated in the *k*-NN algorithm. This equation is given in Equation (20).

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{f=1}^{m} w_f \times (x_{if} - x_{jf})^2} \quad (20)$$

In Equation (20), the distance between sample-i and sample-j is calculated. w_f donates the weight of feature-f. In the same way, x_{if} donates the value of sample-i at feature-f and m donates the number of features.

Table 12: Brief Presentation of the Algorithm 1

Algorithm 1
Step 0: Set Parameters Step 1: Solve Subproblem 1 (Neighborhood Problem) : Use standard *k*-NN with weighted distance measure, Step 2: Solve Subproblem 2 (Weighting Problem): Use FOA to get best weight set, Step 4: Stop: The different *k* value is not obtained

The brief presentation of the Algorithm 1 is given in Table 12. The basic steps of the algorithm have been simply shown in Table 12. The flowchart in which all the steps of the algorithm are shown in detail has been presented in Figure 18.

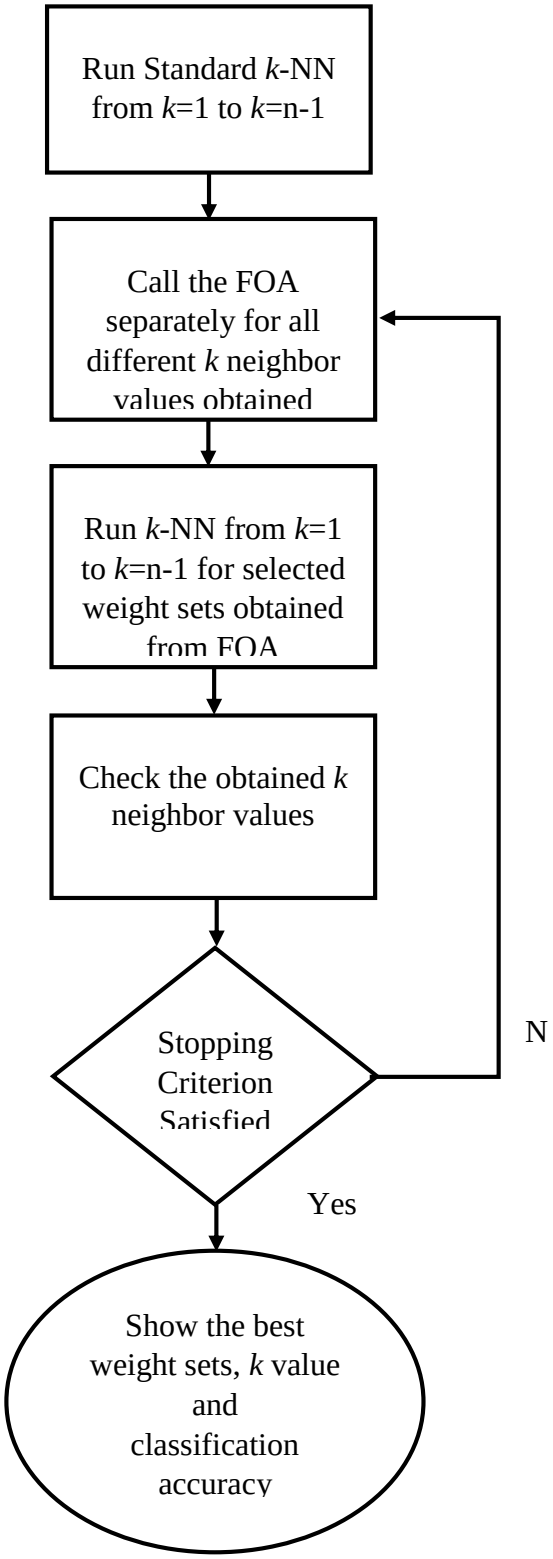

Figure 18: Flowchart of the Algorithm 1

The pseudo-code showing the fundamental processes of the Algorithm 1 is given in Table 13.

Table 13: Pseudo-code of Algorithm 1

Pseudo-Code of Algorithm 1

Step 1: *Standard K-nn:* Set $\mathbf{w} = [1\ 1\ 1\ \ldots\ldots 1]_{1\times m}$ and apply standard *k*-NN algorithm and determine optimum *k* values

Step 2: *Call FOA:* Use FOA algorithm and find optimum weight sets for selected *k* values

```
for i=1: # of selected k values obtained the previous step
    for j=1:5
        Define specific parameters for data sets
        Implement the steps of the FOA algorithm
        Keep results
    end
end
```

Step 3: *K-nn with weighted distance measure:* Apply standard *k*-NN algorithm with calculated distance matrix by using best weight set and determine best *k* values which classification accuracy is highest

```
for i=1: # of selected weight sets
    Apply k-NN with weighted distance measure
    Keep results
end
```

Step 4: *Stop Criteria:* Check stop criterion

Stop Criterion: The obtained best *k* values in step 3 are the same with obtained *k* values previous iterations

Checking: Go to step 5 if stop criterion is satisfied, if not go to step 2 with selected best *k* values.

Step 5: *Results:* Show best **w** weight set, best *k* value and best classification accuracy

4.1.1. Implementation Details of Algorithm 1

With a numerical example, the Algorithm 1 becomes more clear. For example, if we study on a data set of 100 samples, then the standard *k*-NN algorithm is used from *k* = 1 to *k* = n-1 where n is the number of samples in the data set. First of all, the distances between samples are calculated using the weighted Euclidean distance function. At this stage, the weights are assumed to be 1 for all features. The weight matrix is given in Equation (21).

The optimum *k* values for the given weight values is found in the result of the subproblem 1. The *k* values found may be more than one.

$$w = [1\ 1\ 1\ \ldots\ldots 1]_{1\times m} \tag{21}$$

In the subproblem 2, FOA is applied to find best weight set for the *k* value found in the previous step is applied. FOA is used in this step because of the difficulty of solution space.

The parameters used in the FOA vary according to the data set working on. However, some parameters are used as the same for all data sets. The *initial population, area limit, life-time parameters,* and *transfer rate* for all data sets were set to be the same. These parameters were set *initial solutions, weight values,* as *1* for all features, *space limit=30, life limit=6* and *transfer rate=0.1*. For the FOA algorithm used in the subproblem 2, the stopping criterion is defined as the no change of the optimal weight values throughout 100 iterations.

At the end subproblem 2, **w** that maximizes the classification accuracy for the selected *k* value are founded. If more than one *k* values were obtained in the subproblem 1, subproblem 2 is solved separately for each *k* values. FOA is applied five times for every *k* values and the weight values with the highest

classification accuracy are selected as results of subproblem 2. The optimum weight values obtained at the end of this process are evaluated as inputs of the next step(subproblem 1).

Subproblem 1 is solved again by using the best weight sets obtained from subproblem 2. The standard *k*-NN algorithm is run separately for the weight values with the highest classification accuracy obtained in the previous step(subproblem 2). The accuracy of classification from k=1 to k=n-1 is calculated by using the weighted distances between the samples. If the best weight values obtained in the previous step(subproblem 2) are more than one, the distances between samples and the classification accuracy are calculated separately for each **w** values.

The *k* values that maximize the classification accuracy obtained as a result of this step(subproblem 1) are important to decide whether the algorithm will continue or not. If the obtained best *k* values are a subset of the previously obtained *k* values, the stop criterion of the algorithm is provided and the algorithm stops. If the *k* values are not a subset of the *k* values previously obtained, the algorithm returns to the search for the optimum weight values for the *k* values not obtained before, so that the algorithm runs until the stop criterion is ensured. When the stopping criterion is satisfied and the algorithm stops, the best weight set **w**, best *k* value and best classification accuracy are obtained.

4.2. Algorithm 2

In the Algorithm 2 for *k*-NN, the goal is to optimize both *k* value and the feature weight set, **w**, which is the same as the previously proposed the Algorithm 1 to optimize the classification accuracy. However, unlike the previous algorithm, in this algorithm, it is aimed to optimize the *k* value and the feature weight set, **w**, simultaneously not in an iterative scheme.

$$\arg\max\left[\mathrm{f}(k,\mathbf{w})=\sum_{\mathrm{i}=1}^{\mathrm{n}}\delta\left(\mathrm{c}(\mathbf{x_i}),\mathrm{c}_{(k,\mathbf{w})}{}'(\mathbf{x_i})\right)/\mathrm{n}\right] \qquad (FW)$$

The classification accuracy function optimized using the Algorithm 2 has been presented in problem (FW). Here, the FOA is beneficial to solve this problem. As mentioned earlier, every candidate solution in FOA is called a tree. The tree structure used to solve this problem is shown in Equation (22).

$$\mathrm{Tree}:[\mathrm{w}_1,\mathrm{w}_2,\mathrm{w}_3,\mathrm{w}_4,\ldots,\mathrm{w}_\mathrm{m},\mathrm{k\ value},Age] \qquad (22)$$

In this tree structure, unlike the tree structure in the Algorithm 1, the *k* value is also added as a variable into the tree. With introducing the *k* value into the tree, both *k* and **w** are optimized into FOA. All weights are assigned as 1 in the initial solution space used in the FOA. *K* values to be assigned to the tress in the initial solution space are obtained using the standard *k*-NN.

Algorithm 2 basically consists of 2 steps. First, initialization step, *k* values are obtained by applying the standard *k*-NN algorithm for *k* = 1 to assign to the trees in the initial solution space. This step is implemented just one time in Algorithm 2 for all data sets. Second, the FOA with previously gotten *k* value to use in trees in initial solution is called to solve the feature weighted *k*-NN problem. The flowchart of this algorithm is given in Figure 19.

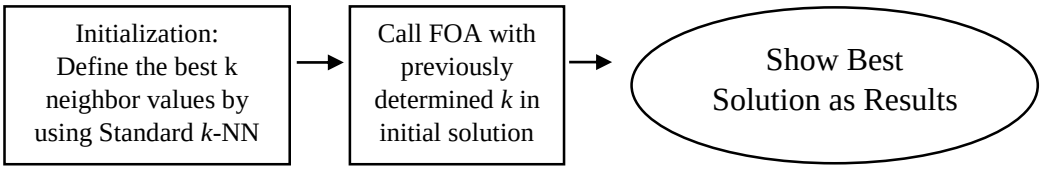

Figure 19: Flowchart of the Algorithm 2

By using the standard *k*-NN algorithm in initialization step, the best *k* values are found and sorted at the beginning of the Algorithm 2, in the case where all the features have the same relevance level. From obtained *k* values, set of *k* are selected starting from the best value as much as the number of initial solution parameter in the FOA. The FOA algorithm is called after the best *k* values have been determined. The initial forest is formed by combining the feature weight values generated with the assumption that all features are initially at the same level of interest with the *k* values determined. A representative initial solution, the forest, is shown in Equation (23) under the assumption that the population limit is 5 and the best *k* values obtained are 3, 7, 1, 9 and 5, respectively. The FOA is called to optimize simultaneously the feature weights and *k* value.

$$Initial\ Forest: \begin{bmatrix} 1,1,1,1,\dots,1,3,0 \\ 1,1,1,1,\dots,1,7,0 \\ 1,1,1,1,\dots,1,1,0 \\ 1,1,1,1,\dots,1,9,0 \\ 1,1,1,1,\dots,1,5,0 \end{bmatrix} \tag{23}$$

The most important consideration for this algorithm when applying FOA is that the feature weights are continuous variables, however, the *k* values are discrete variables. Due to this difference, the entire FOA structure should be updated by considering this difference. Especially, the local seeding and global seeding steps must be carefully designed.

The pseudo-code of proposed Algorithm 2 is presented in Table 14.

Table 14: Pseudo-code of Algorithm 2

Pseudo-Code of Algorithm 2

Step 0: *Define parameters*; Area Limit, Life Time, LSC, GSC, Transfer Rate
Step 1: *Standard K-nn:* By using standard *k*-NN, define best *k* values to implemet initial forest for FOA
Step 2: *Initialization of FOA:* Create the initial forest with random trees which include feature weight set and *k* with zero age,
Step 3: *Iteration:* While stop condition is not satisfied do,
Step 3.1: Local seeding to trees with zero age
for i=1:LSC,
Select a variable randomly from the selected tree,
if : selected variable is *k* value,
Add 1 or -1 to the *k* value,
else
Add a small random value to the selected variable,
end
end
Increase by 1 the age of all other trees except the new ones,
Step 3.2: Calculate fitness value of all trees and sort them according to fitness value,
Step 3.3: Population Limiting
Remove the trees that have bigger age than the life time parameter and send them to the candidate population,
Remove the trees exceeding the area limit from the end of forest and send them to the candidate population,
Step 3.4: Global Seeding
Choose trees from candidate population according to transfer rate parameter,
for i=1:number of selected trees,
Select randomly "GSC" variables of selected tree,
Change the selected variables with randomly generated value which must be in the selected variable range,
Make the age of selected tree zero and add the tree to the forest,
end
Step 3.5: Calculate fitness value of all trees and sort them according to fitness value,
Step 3.6: Make the age of best tree zero
Step 3.7: Population Limiting
Remove the trees that have bigger age than the life time parameter and send them to the candidate population,
Remove the trees exceeding the area limit from the end of forest and send them to the candidate population,
Step 4: *Results:* Show the best tree as a result

4.3. Application Results

The proposed algorithms, which are described in detail, has been applied to 6 different data sets. These data sets are Bupa, Dermatology, Glass, Heart, Iris and Pima data sets from UCI Machine Learning Repository. Properties of these data sets are given in Table 15.

Table 15: Properties of Data Sets

	# of samples (n)	# of features	# of class
Bupa	345	6	2
Dermatology	366	33	6
Glass	214	9	7
Heart	270	13	2
Irıs	150	4	3
Pima	768	8	2

The FOA is used with the same parameters in both algorithms. The parameters used in FOA for the data sets are as in Table 16. The leave-one-out cross validation approach has been used in training and test set separation for all *k*-NN applications used in this study.

Table 16: FOA Parameters

Data Set	Area Limit	Life Time	Transfer Rate	LSC	GSC
Bupa	30	6	0.1	1	2
Dermatology	30	6	0.1	7	10
Glass	30	6	0.1	2	3
Heart	30	6	0.1	2	4
Irıs	30	6	0.1	1	1
Pima	30	6	0.1	2	2

4.3.1. Results of Algorithm 1

The first step of Algorithm 1 is Neighborhood Problem(Subproblem 1). In subproblem 1, first, optimal *k* value is searched by using standard Euclidean distance function. The weighting problem, which is the second step of the Algorithm 1, is applied for every *k* values obtained. FOA is used for weighting problem. The FOA parameters used for all data sets are shown in Table 16. These parameters are the parameters used in Ghaemi and Feizi-Derakhshi (2014) study.

The *k* values with the best classification accuracy obtained for the data sets at the end of the first solution of the subproblem 1 are given in Table 17.

Table 17: Best *k* neighbor values at end of the first step

	Bupa	Dermatology	Glass	Heart	Iris	Pima
k	29	5,6	5	57	30,31,36	19

In the next step, weighting problem (subproblem 2) is solved by using FOA. For all data sets, FOA is initialized 5 times and classification accuracy results as a percentage are shown in Table 18. The best classification accuracy for each data set is given in bold.

Table 18: Classification Accuracy at the end of the second step

	k	Int. 1	Int. 2	Int. 3	Int. 4	Int. 5
Bupa	29	71.59	71.59	**72.17**	71.30	**72.17**
Dermatology	5	**99.18**	**99.18**	**99.18**	98.91	98.91
	6	98.91	**99.18**	**99.18**	**99.18**	**99.18**
Glass	5	73.83	**75.13**	**75.13**	74.30	74.30
Heart	57	**88.15**	87.04	87.41	87.78	86.67
Irıs	30	**97.33**	**97.33**	**97.33**	**97.33**	**97.33**
	31	**97.33**	**97.33**	**97.33**	**97.33**	96.66
	36	96.67	96.67	96.67	96.67	96.67
Pima	19	79.95	**80.34**	79.56	80.08	**80.34**

The weight values obtained from the first solution of subproblem 2 are used to calculate feature weighted distance function in the subproblem 1. Feature weights with the best classification accuracy are used for all data sets. This means that 2 different *k*-NN with weighted distance function for Bupa, 7 different for Dermatology, 2 different for Glass, 1 for Heart, 9 different for Iris and 2 different for Pima will be applied.

In step 3, subproblem 1 is solved again. *k*-NN with feature weighted distance function was applied using selected weights for each data set and the results are presented in Table 19. The *k* values that have not been obtained previously are shown as bold in Table 19.

Table 19: Best *k* neighbor values at end of the second solution of Subproblem 1

	Bupa	**Dermatology**	**Glass**	**Heart**	**Iris**	**Pima**
k	29	5,6	**3**,5	57	**10,11,12,13,14,17**	19

Here, the first iteration of the Algorithm 1 is completed. At this stage, the decision whether the algorithm will continue or not is made by checking stopping condition. Table 19 is examined, it is clear that only a new *k* neighbor value is obtained for Glass and Iris data sets and the old *k* values are repeated for the other data sets. This means that the algorithm stops here for all data sets except Glass and Iris data sets.

In the next step, the subproblem 2 is solved a second time, the optimal feature weight values for the Glass data set with $k = 3$ and for Iris data set with k=10,11,12,13,14, and 17 are searched by using the FOA algorithm. As a result of this searches, the classification accuracy values obtained are shown in Table 20.

The best classification accuracy value was obtained from two different initialization for Glass data set and three different initialization for Iris data set. This means that 2 different feature weight sets for Glass and 3 for Iris are obtained. These five feature weight sets are used as input to the next subproblem 1.

Table 20: Classification Accuracy at the end of the fourth step

	k	Int. 1	Int. 2	Int. 3	Int. 4	Int. 5
Glass	3	75.13	75.70	**77.10**	**77.10**	76.64
Iris	10	98.00	98.00	98.00	98.00	98.00
	11	98.00	98.00	98.00	98.00	98.00
	12	98.00	98.00	98.00	98.00	98.00
	13	98.00	98.00	97.33	96.67	97.33
	14	98.00	98.00	**98.67**	**98.67**	98.00
	17	98.00	**98.67**	98.00	98.00	98.00

In step 4, the subproblem 1 is resolved a third time, *k*-NN with feature weighted distance function is applied to Glass and Iris data set using the weight sets obtained in the previous step. As a result of this implementation, the highest classification accuracy is obtained for $k = 3$ for Glass data set and k=14 and 17 for Iris data set.

Here, the first iteration of the Algorithm 1 is completed and it is time to check stopping condition. *K* values obtained for both data sets are the subset of the previously obtained *k* values. This means that the Algorithm 1 ends for both Glass data set and Iris data set.

Table 21: Best Classification Accuracy at the end of Algorithm 1

	k	Best Classification Accuracy
Bupa	29	72.17
Dermatology	5,6	99.18
Glass	3	77.10
Heart	57	88.15
Iris	14,17	98.67
Pima	19	80.34

The classification accuracy values obtained by using the Algorithm 1 for all data sets are given in Table 21.

In Table 22, the classification accuracies obtained are compared with the given values in Ghaemi and Feizi-Derakhshi (2014) study. The best classification values are given in bold. Brief information abaout the methods shown in the Table 22 is presented in Appendix B.

Table 22: Algorithm 1 vs. Some Other Algorithms

	INN		SSMA		SSMA-DEPGFW		1 PA DECS		IPADECS-DEFW		TSKNN		FOA		ALGORITHM 1	
Data Sets	**ACC**	**SD**	**ACC**	**SD**	**ACC**	**SD**	**ACC**	**SD**	**ACC**	**SD**	**ACC**	**SD**	**ACC**	**SD**	**ACC**	**SD**
Bupa	61.08	6.88	62.70	8.47	67.41	7.96	65.67	8.48	67.25	5.27	62.44	7.90	67.60	0.43	**71.76**	0.3.5
Derma.	95.35	3.45	95.10	5.64	94.02	4.31	96.18	3.01	96.73	2.64	96.47	4.01	97.37	0.31	**99.18**	0.00
Glass	73.61	11.90	68.80	8.19	73.64	8.86	69.09	11.13	71.45	11.94	76.42	13.21	**78.62**	0.7	76.35	0.76
Iris	93.33	516	96.00	4.42	94.67	4.99	94.67	4.00	94.67	4.00	94.00	4.67	96.66	0.00	**98.67**	0.00
Pima	70.33	74.30	74.23	4.01	73.23	5.43	76.84	4.67	71.63	7.35	75.53	5.85	71.11	0.00	**80.20**	0.16
Heart	77.04	8.89	83.70	10.1	85.19	8.11	83.70	9.83	80.74	9.19	81.48	6.42	81.35	0.37	**87.41**	0.52

As a result, when the classification accuracy values are examined, it is seen that the proposed the Algorithm 1 yields successful results and it has succeeded in optimizing both the *k* value and the feature weight set **w** iteratively.

4.3.2. Results of Algorithm 2

In this algorithm, the standard *k*-NN is applied initially and the best 30 neighbor values are determined. These determined 30 neighbor values will be used as input for the FOA to be used in the next stage. After determining best neighborhood values, the second stage, which the FOA is used, is proceeded.

The distinction of the second phase from the previous implementations of FOA is the difference in the structure of the trees that form the initial forest. In addition to feature weights and age parameters, *k* values are stored in these trees. In Equation (22), there is a representative example of a tree. Trees in the initial population have weight sets which form ones, mean that all features have same relevance level, and *k* value. In Figure 21, the first tree in the initial population for the Bupa data set is presented as an example to get rid of the question that may arise.

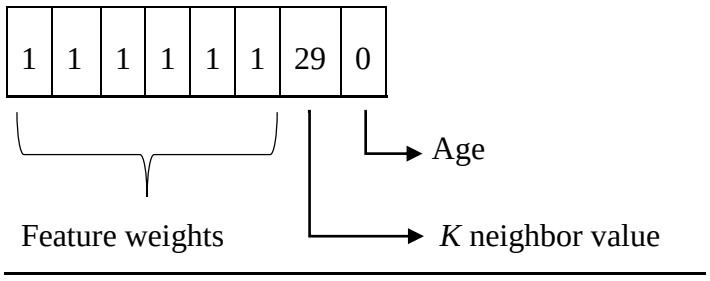

Figure 20: First Tree of Initial Forest of BUPA Data Set

As shown in Figure 20, weights for all features are set as 1 and ages are set as 0 for trees in the initial forest. *K* neighbor values are assigned according to the values obtained from the standard *k*-NN, which is the first step of the Algorithm 2. In the first step, the best 30 neighbor values are determined using the standard *k*-NN algorithm.

In the second step, the FOA is applied using the neighbor values obtained in the first step in creating the initial forest. The parameters used in FOA for all data sets are shown in Table 16. The FOA algorithm for each data set is run 10 times and the best results which have best classification accuracy are presented in Table 23.

Table 23: Best Classification Accuracy at the end of Algorithm 2

	k	**Best Classification Accuracy**
Bupa	16	72.75
Dermatology	5,6,8	99.18
Glass	3	77.58
Heart	47,57,64	88.85
Iris	15	98.67
Pima	23	80.60

As it can be clearly seen from Table 23, the proposed Algorithm 2 gets successful results in considering the classification accuracy obtained and optimizes both the *k* value and the feature weight values to improve the classification accuracy.

4.3.3. Comparison of the Results of Two Proposed Algorithms

Classification accuracy is considered as a performance criterion. The best classification accuracies obtained for 6 data sets will be considered to compare these two algorithms. The best classification accuracies, for two proposed algorithm, obtained for all data sets are presented in Table 24.

Table 24: Comparison of Two Proposed Algorithms

	Algorithm 1	**Algorithm 2**
Bupa	0.7217	**0.7421**
Dermatology	0.9918	0.9918
Glass	0.7710	**0.7758**
Heart	0.8815	0.8815
Iris	0.9867	0.9867
Pima	0.8034	**0.8060**

It is possible to say that the Algorithm 2 achieves more successful results when the best classification accuracy values are taken into consideration. Although two algorithms have reached the same classification accuracy in 3 out of 6 data sets, the Algorithm 2 has achieved a higher classification accuracy on the remaining 3 data set.

When Table 22 is examined, it is obvious that the Algorithm 1 shows superior performance in all data sets except Glass data set. Therefore, it would be sufficient to compare Algorithm 1 to evaluate the performance of Algorithm 2 in terms of average classification accuracy and standard deviations. In Table 25, this comparison is presented and best classification accuracy is given in bold.

Table 25: Algorithm 1 vs. Algorithm 2

	Algorithm 1		Algorithm 2	
Data Sets	ACC	SD	ACC	SD
Bupa	71.76	0.35	**73.97**	0.11
Dermatology	**99.18**	0.00	**99.18**	0.00
Glass	76.35	0.76	**77.15**	0.14
Irıs	**98.67**	0.00	**98.67**	0.00
Pima	80.20	0.16	**80.24**	0.26
Heart	87.41	0.52	**87.96**	0.18

It is seen that the results shown in Table 25 are almost the same as results in Table 24. Although the Algorithm 2 has achieved a higher classification accuracy in the remaining 4 data sets, two algorithms have reached the same classification accuracy in 2 out of 6 data sets.

Although the classification accuracy obtained by Algorithm 2 in Glass data set is better than Algorithm 1, it is not well than in the study by Ghaemi and Feizi-Derakhshi (2014). This means that Algorithm 2 has achieved more successful results than Algorithm 1 in 4 out of 6 data sets and in 5 out of 6 data sets in other studies.

CHAPTER 5

CLASS DEPENDENT FEATURE WEIGHTED *k*-NN

In problem 1 described in Chapter 4, feature weights were treated the same for each class. This implies that each feature has the same value for all classes. There are many examples in which features have actually different meanings for each class in classification problems in real life. In this chapter, the problem that the feature weights vary by class has been handled to improve the classification accuracy and to obtain new information about the class-based significance of the features. Class Dependent Feature Weighted *k*-NN Problem (CDFW *k*-NN) is given in problem (CFW).

$$\arg\max\left[\mathrm{f}(k,\mathbf{w_c}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \delta\left(\mathrm{c}(\mathrm{x_i}), \mathrm{c}_{(k,\mathbf{w_c})}{}'(\mathrm{x_i})\right)/\mathrm{n}\right] \qquad (CFW)$$

Unlike the Problem 1 described in Chapter 4, the weight set consists of a different vector for each class in this problem. This problem which a classification accuracy function is given in problem (CFW).

Breast Cancer data set from real life problems is a good example of how features have different importance for different classes. This data set contains information about the classification of the tumor and the results of 9 medical examinations of the patient. In this data set, it appears that the tumor size has a different level of relevance for recurrence or non-recurrence classes (Marchiori, 2013).

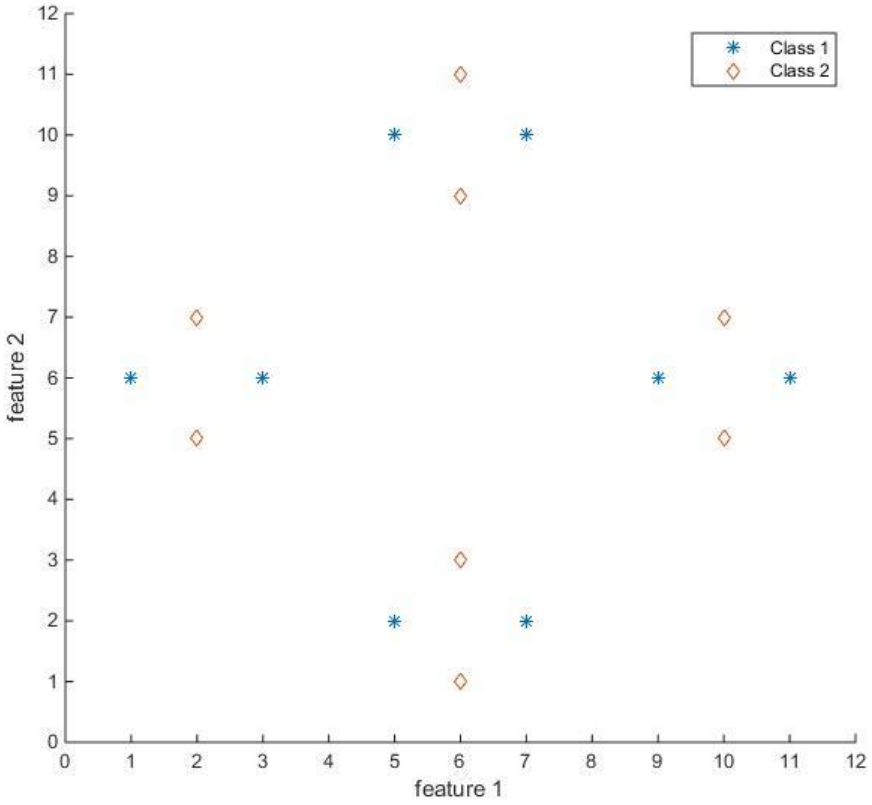

Figure 21: Toy Example

In Figure 21, a sample synthetic data set scatter graph is given. The data set consists of two different classes and 16 different samples and each samples having two features.

Three different algorithms, standard *k*-NN, feature weighted *k*-NN and class dependent feature weighted *k*-NN (CDFW *k*-NN), applied on this data set with *k*=1. The leave-one-out method was chosen as the cross validation method. Standard *k*-NN can not classify any point correctly.

If the data set is handled according to the feature weighted *k*-NN problem, the weight of one feature is set to high value and the other one set to very low value. If the weight of the feature indicated by the x-axis is assigned as high value, 4

the samples in class 1 are incorrectly classified, whereas the 4 samples in class 1 and all sample in class 2 are correctly classified. If the weight of feature represented by the y-axis is assigned as high value, 12 out of 16 samples (except 4 points in class 2) are correctly classified. This means that the classification accuracy is 75% in the case of the feature weighted *k*-NN problem.

Finally, if class dependent feature weighted *k*-NN is used, the weight of the feature 1 is set high value than the weight of the feature 2 for class 2, and the weight of the feature 2 is set to high value than the weight of the feature 1 for class 1. All samples are correctly classified and the classification is performed with 100% success.

5.1. Feature Weights

In previous applications, feature weights were considered the same for each class. The representative example of a feature weight set is shown previously in Equation (21) in Chapter 4. As clearly shown in the example, the weight set is a vector with one weight value for each feature. However, in class dependent feature weighting approach, different feature weights are used for each class. In this case, the sample feature weights set shown in Equation (21) is updated as in Equation (24).

$w_f(c_i)$ =weight of feature f for class i.

$$\mathbf{w} = \begin{bmatrix} w_{11} w_{12} \dots w_{1m} \\ w_{21} w_{22} \dots w_{2m} \\ \dots \dots \dots \dots \dots \dots \\ w_{c1} w_{c2} \dots w_{cm} \end{bmatrix}_{c \times m} \tag{24}$$

5.2. Distance Measure

The feature weighted Euclidean distance function is used as the distance measure. The feature weighted Euclidean distance function used for the same set of weights for each class is given in Equation (20) in Chapter 4. However, since the feature weights to be used here are class dependent, the feature weighted Euclidean function has been changed. Adapted feature weighted Euclidean distance function to this problem is given in Equation (25).

$$c(i) = \text{class of sample i,}$$

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{f=1}^{m} w_f(c(j)) \times (x_{if} - x_{jf})^2} \qquad (25)$$

The most important point in the feature weighted Euclidean distance function shown in Equation (25) is the weight value used in the calculation. The feature weight set of the class to which the point selected from the training set belongs is used while calculating the distance between a selected point from the test set and selected point from the training set.

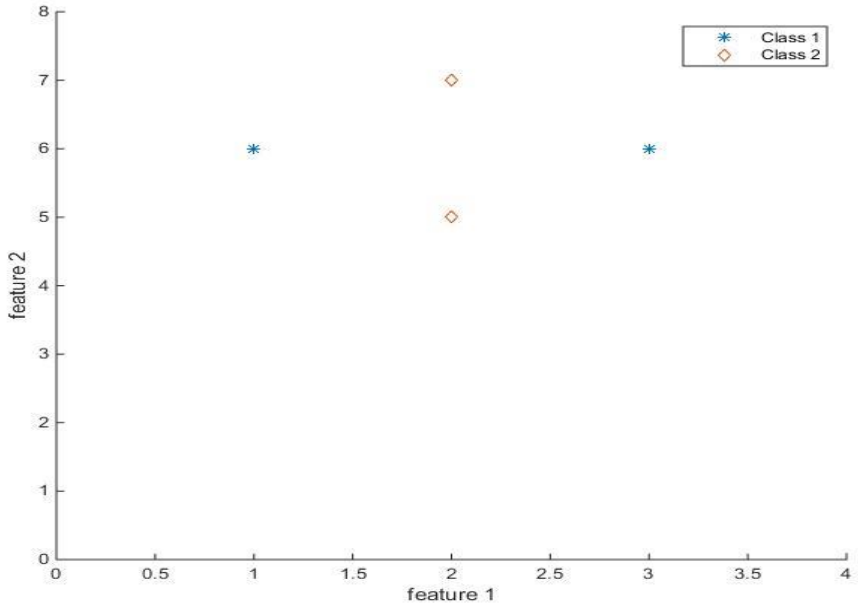

Figure 22: Selected Some Points from Toy Example in Figure 21

To illustrate the distance calculation more clearly, selected 4 points from Figure 23 is shown in Figure 22. These points are called (1,6), (2,5), (2,7) and (3,6) and they are called $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ and $\mathbf{x_4}$ respectively. As it can be seen from Figure 24, $\mathbf{x_1}$ and $\mathbf{x_4}$ belong to class 1 and $\mathbf{x_2}$ and $\mathbf{x_3}$ belong to the second class. One sample of the weight matrix for this problem is also given in Equation (26).

$$\mathbf{w} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_{2\times 1} \tag{26}$$

Within the framework of this information;

The distance between $x_1$and x_2 is calculated as in Equation (27).

$$\mathrm{d}(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{f=1}^{2} \mathrm{w}_{\mathrm{c}(2)\mathrm{f}} \times (\mathrm{x}_{1\mathrm{f}} - \mathrm{x}_{2\mathrm{f}})^2}$$

$$= \sqrt{\mathrm{w}_{\mathrm{c}(2)1} \times (\mathrm{x}_{11} - \mathrm{x}_{21})^2} + \sqrt{\mathrm{w}_{\mathrm{c}(2)2} \times (\mathrm{x}_{12} - \mathrm{x}_{22})^2} \quad (27)$$

$$= \sqrt{1 \times (1-2)^2} + \sqrt{0 \times (6-5)^2} = 1$$

In the same way, distance between $\mathbf{x_1}$and $\mathbf{x_3}$ is calculated as in Equation (28);

$$= \sqrt{1 \times (1-2)^2} + \sqrt{0 \times (6-7)^2} = 1 \quad (28)$$

Finally, distance between $\mathbf{x_1}$and $\mathbf{x_2}$ is calculated as in Equation (29);

$$= \sqrt{0 \times (1-3)^2} + \sqrt{1 \times (6-6)^2} = 0 \quad (29)$$

When the CDFW *k*-NN algorithm is applied with the 1-nearest neighbors, $\mathbf{x_1}$ is assigned to class 1 and this is a true assignment.

5.3. Forest Optimization Algorithm Application

FOA is used as a solution method in the problem (CFW) as well as in the previously proposed algorithms. The problem (CFW) solution method by using FOA is called Algorithm 3.

The most important part of applying the FOA to problem (CFW) is to determine the tree structure correctly and effectively. In this frame, the weight matrix is transformed into a vector and used in the FOA. The weight set for the vector used in the FOA is shown in Equation (30).

$$\mathbf{w} = [w_{11}, w_{12}, \dots w_{1m}; w_{21} w_{22} \dots w_{2m}; \dots \dots \dots; w_{c1}, w_{c2}, \dots w_{cm}] \qquad (30)$$

Trees in FOA have weights and the age, which is a parameter for that tree. Considering this, a tree that forms the initial forest under the assumption that all features initially have the same level of relevance is shown in Equation (31).

$$\mathbf{w} = [1,1, \dots 1; 1,1 \dots 1; \dots \dots \dots; 1,1, \dots 1; 0] \qquad (31)$$

The trees that form the initial forest were created with this structure for all data sets. How many weights are in the trees that form the initial forest is determined by the number of features and the number of classes in the data set. For example, for the IRIS data set, each sample has 4 features and 3 different classes in data set, the trees forming the initial forest have 12 weight values and 1 age parameter, so each tree has 13 elements.

The parameters used in the FOA for each data set are given in Table 25.

Table 26: Parameters of FOA for CDFW *k*-NN

Data Set	Area Limit	Life Time	Transfer Rate	LSC	GSC
Bupa	30	6	0.1	3	6
Dermatology	30	6	0.1	40	80
Glass	30	6	0.1	13	26
Heart	30	6	0.1	6	12
Irıs	30	6	0.1	3	6
Pima	30	6	0.1	4	8

It is recommended that to set the *LSC* parameter to 1/5 of the dimension of each data set in Ghaemi and Feizi-Derakhshi (2016). In this view, the *LSC* parameter is set and the *GSC* parameter is set to 2 times the *LSC* parameter. The stopping criterion was determined as no change in the best tree over 100 iterations.

5.4. Results of Algorithm 3

The Algorithm 3 is also applied to, as in the previously proposed algorithms, the 6 different data sets presented in Table 15 in Chapter 4. FOA was used to solve the problem and 5 initializations were made for each data set.

The best classification accuracy values obtained from the application of this algorithm for each data sets are shown in Table 27.

Table 27: Best Classification Accuracy of Algorithm 3

	Best Classification Accuracy (%)
Bupa	73.04
Dermatology	98.91
Glass	80.37
Heart	87.78
Iris	98.00
Pima	76.17

To better understand the success of the classification accuracy obtained, average classification accuracies and standard deviations are compared with the classification accuracy of FW *k*-NN application with same data sets in Ghaemi and Feizi-Derakhshi (2014).

Table 28: CDFW *k*-NN vs FW *k*-NN

	CDFW *K*-nn		**FW *k*-NN**	
	ACC	**SD**	**ACC**	**SD**
Bupa	**72.87**	0.14	67.60	0.43
Dermatology	**98.80**	0.13	97.37	0.31
Glass	**79.35**	1.10	78.62	0.7
Heart	**86.22**	1.00	81.35	0.37
Iris	**97.87**	0.27	96.66	0.00
Pima	**75.21**	0.64	71.11	0.00

The best classification accuracy values for each data set are marked in bold in Table 28. The Algorithm 3 has shown a much better performance than the FW *k*-NN algorithm, as clearly shown in Table 28.

As mentioned earlier, the purpose of the CDFW *k*-NN problem is to improve the classification accuracy, as well as to determine the different relevance level of features for different classes.

Table 29: Sample Weight Set for IRIS Data Set

	Feature 1	Feature 2	Feature 3	Feature 4
Class 1	1.00	0.51	0.15	0.33
Class 2	0.80	0.53	0.84	0.31
Class 3	0.45	0.01	0.65	1.00

In Table 29, there is a sample class dependent weight set found for the Iris data set. Each row shows the weight set for each class, and each column shows the feature weight for each class. Having different feature weights for each class provides new information about relevance level of each feature for each class. This new information can be used to determine which features are more relevant for which classes. For example, feature 1 has a very high level of relevance for class 1, however, for class 3, relevance level of feature 1 is low. Relevance level of feature 2 is moderate for class 1 and class 2, however very low for class 3, even irrelevant. For feature 3, the opposite situation is true. Feature 3 has a high level of relevancy for class 3, and a low level of relevance for class 1. Feature 4 has a very high level of relevance for class 3, and has a low level of relevance for class 1 and class 2.

The weight set obtained in the solution of the CDFW problem actually has shown function of class dependent feature selection. A threshold can be determined for the feature weights obtained and feature weights which are below this level may not be taken into account for related class.

Table 30: Performance of Alg.1, Alg.2 and Alg.3

	Best Classification Accuracy (%)		
	Alg. 1	**Alg. 2**	**Alg. 3**
Bupa	72.17	**74.21**	73.04
Dermatology	**99.18**	**99.18**	98.91
Glass	77.10	77.58	**80.37**
Heart	**88.15**	**88.15**	87.78
Iris	**98.67**	**98.67**	98.00
Pima	80.34	**80.60**	76.17

In Table 30, the results of the Algorithm 3 are compared with the results obtained from Algorithm 1 and 2. Algorithm 1 and Algorithm 2 are more successful than Algorithm 3 in terms of classification accuracy. The most basic reason of this situation is that in Algorithm 1 and Algorithm 2, besides feature weights, *k* value also is considered as part of the optimization problem. However, the *k* value in the Algorithm 3 has been accepted as 1 and only the feature weights have been optimized to maximize the classification accuracy. In future studies on the CDFW problem, algorithms optimizing both the *k* value and the feature weight set, **w**, can be proposed to improve classification accuracy.

CHAPTER 6

CONCLUSION

In this thesis, the feature weighted *k*-Nearest Neighbor (*k*-NN) problem is studied. Firstly, the terms used in the study are explained with examples in order to clarify the study and the details of the problem and the studies in the literature made about this problem have been mentioned. The stages of the NN problem are explained and the feature weighted *k*-NN problem is presented in detail.

The feature weighted *k*-NN problem is a nonlinear problem because of weighted Euclidean distance function. This makes the solution of the feature weighted *k*-NN problem in traditional mathematical ways almost impossible. Therefore, Forest Optimization Algorithm(FOA), an evolutionary algorithm that has been reported to be successful in solving nonlinear problems, has been used as a fundamental solution tool in this thesis. The FOA algorithm is described in detail with all its parameters and processes. The performance of the FOA has been measured and reported with test functions frequently used in the literature for evaluation.

The feature weighted *k*-NN problem is expressed as a function depending on the *k* value and the set weight **w**. With this definition, the feature weighted *k*-NN has become an optimization problem with 2 parameters. The purpose of this optimization problem is to maximize the accuracy of classification. Two different algorithms have been proposed to solve this described problem.

In the first algorithm, called Algorithm 1, the feature weighted *k*-NN problem is divided into 2 subproblem. These subproblems are called Neighborhood

Problem and Weighting Problem, respectively. Algorithm 1 is an iterative algorithm between these two subproblems. In Neighborhood Problem, the Standard *k*-NN algorithm is implemented using feature weighted Euclidean distance function. The goal in this problem is to find the *k* values that maximize the classification accuracy using the given weight sets. After the best *k* values are determined, subproblem 2 (the weighting problem) is handled. The goal in the weighting problem is to find the best weight sets, which maximize the classification accuracy, for the *k* values obtained as a result of the neighborhood problem. The weighting problem is applied for each *k* values separately and the best set of weights for each *k* values is determined. The solution space of the weighting problem is a nonlinear structure. For this reason, FOA is used to solve the weighting problem. FOA is initialized to 5 for each *k* value and feature weight sets with the highest classification accuracy are recorded as the results of the weighting problem. The obtained weight sets are used as input in the neighborhood problem. By using these weight sets, neighbor problem is solved again. This iterative structure continues until the *k* values obtained as a result of last solved neighborhood problem become a subset of the previously obtained *k* values.

In the second algorithm, *k* value and **w** feature weight set are optimized simultaneously in the feature-weighted *k*-NN to maximize classification accuracy. In the same reason as the weighting problem, FOA is used to solve the problem.

The feature weighted *k*-NN problems on 6 data sets are solved using Algorithm 1 and Algorithm 2. The results of the two proposed algorithms are compared both with each other and with the results in the literature. As a result of these comparisons, it has been seen that the two proposed algorithms have performed very successful results.

In this thesis, the feature weighted *k*-NN problem is redefined as a different problem using class dependent feature weights. This problem is defined as class dependent feature weighted *k*-NN(CDFW *k*-NN) problem. There are two main purposes of defining this problem; first, to increase the classification accuracy and second, to determine features relevance level for each class, by allowing them to take different weights for different classes. FOA is used to solve this defined problem. The classification accuracy values obtained are recorded and compared with the results reported in the literature. The results show that the proposed model is successful.

In this thesis, in summary, the feature weighting problem is applied on *k*-NN. In this framework, two different problems are defined, feature weighted *k*-NN and class dependent feature weighted *k*-NN, and algorithms for solving these problems are proposed. Because of the nonlinear structure of these problems, FOA is used in the proposed algorithms. The comparison of the obtained results with the literature shows the success of the proposed algorithms

REFERENCES

Alpaydin, E. (2014). Introduction to machine learning. *MIT press*.

Barboza, F., Kimura, H., & Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, *83*, 405-417.

Batista, G. E., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied artificial intelligence*, *17*(5-6), 519-533.

Belkasim, S. O., Shridhar, M., & Ahmadi, M. (1992). Pattern classification using an efficient KNNR. *Pattern Recognition*, *25*(10), 1269-1274.

Bhatia, N. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*.

Buturović, L. J. (1993). Improving k-nearest neighbor density and error estimates. *Pattern Recognition*, *26*(4), 611-616.

Byers, W. A., & Perone, S. P. (1980). Nearest neighbor rule in weighting measurements for pattern recognition. *Analytical Chemistry*, *52*(13), 2173-2177.

Chaghari, A., Feizi-Derakhshi, M. R., & Balafar, M. A. (2016). Fuzzy clustering based on Forest optimization algorithm. *Journal of King Saud University-Computer and Information Sciences*.

Chan, E. Y., Ching, W. K., Ng, M. K., & Huang, J. Z. (2004). An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern recognition*, *37*(5), 943-952.

Chien, J. T., & Wu, C. C. (2002). Discriminant waveletfaces and nearest feature classifiers for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(12), 1644-1649.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21-27.

Cramer, S., Kampouridis, M., Freitas, A. A., & Alexandridis, A. K. (2017). An extensive evaluation of seven machine learning methods for rainfall prediction in weather derivatives. *Expert Systems with Applications*, *85*, 169-181

Criminisi, A. (2016). Machine learning for medical images analysis. *Medical Images Analysis*, 33, 91-93

de Amorim, R. C. (2016). A survey on feature weighting based K-Means algorithms. *Journal of Classification*, *33*(2), 210-242.

DeSarbo, W. S., Carroll, J. D., Clark, L. A., & Green, P. E. (1984). Synthesized clustering: A method for amalgamating alternative clustering bases with differential weighting of variables. *Psychometrika*, *49*(1), 57-78.

Dialameh, M., & Jahromi, M. Z. (2017). A general feature-weighting function for classification problems. *Expert Systems with Applications*, *72*, 177-188.

Dubitzky, W., Granzow, M., & Berrar, D. P. (Eds.). (2007). *Fundamentals of data mining in genomics and proteomics*. Springer Science & Business Media.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4), 325-327.

Forbes, R. A., Tews, E. C., Freiser, B. S., Wise, M. B., & Perone, S. P. (1986). Development of a novel weighting scheme for the k-nearest-neighbor algorithm. *Journal of Chemical Information and Computer Sciences*, *26*(3), 93-98.

García, S., Cano, J. R., & Herrera, F. (2008). A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, *41*(8), 2693-2709.

Ghaemi, M., & Feizi-Derakhshi, M. R. (2014). Forest optimization algorithm. *Expert Systems with Applications*, *41*(15), 6676-6687.

Ghaemi, M., & Feizi-Derakhshi, M. R. (2016). Feature selection using forest optimization algorithm. *Pattern Recognition*, *60*, 121-129.

Goin, J. E. (1984). Classification bias of the k-nearest neighbor algorithm. *IEEE transactions on pattern analysis and machine intelligence*, (3), 379-381.

Gutierrez-Osuna, R. (2002). Pattern analysis for machine olfaction: a review. *IEEE Sensors journal, 2(3)*, 189-202.

Hattori, K., & Takahashi, M. (1999). A new nearest-neighbor rule in the pattern classification problem. *Pattern recognition*, *32*(3), 425-432.

Heikkonen, J., Varfis, A., Kanellopoulos, I., Wilkinson, G. G., Fullerton, K., & Steel, A. (1997). A method for remote sensing based classification of urban areas. In *Scandinavian conference on image analysis* (pp. 1015-1022).

Hmeidi, I., Hawashin, B., & El-Qawasmeh, E. (2008). Performance of KNN and SVM classifiers on full word Arabic articles. *Advanced Engineering Informatics*, *22*(1), 106-111.

Huang, J. Z., Ng, M. K., Rong, H., & Li, Z. (2005). Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(5), 657-668.

Jain, A. K., & Mao, J. (1991, July). A k-nearest neighbor artificial neural network classifier. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on* (Vol. 2, pp. 515-520). IEEE

Jiang, L., Cai, Z., Wang, D., & Jiang, S. (2007, August). Survey of improving k-nearest-neighbor for classification. In *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on* (Vol. 1, pp. 679-683). IEEE.

Jiang, L., Li, C., Wang, S., & Zhang, L. (2016). Deep feature weighting for naive Bayes and its application to text classification. *Engineering Applications of Artificial Intelligence*, *52*, 26-39.

Jing, J., Shouyong, W., Lan, Y., Delin, Z., Zhaoming, Y., & Changwen, T. (1998). Radar target recognition based on fuzzy clustering. In *Signal Processing Proceedings, 1998. ICSP'98. 1998 Fourth International Conference on* (Vol. 2, pp. 1532-1535). IEEE

Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4), 580-585.

Kira, K., & Rendell, L. A. (1992, July). A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning* (pp. 249-256).

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Lu, Y. J., Hsu, Y. H., & Maldague, X. (1992). Vehicle classification using infrared image analysis. *Journal of transportation engineering*, *118*(2), 223-240.

M. Li, M. M. Crawford, and J. Tian, (2010). "Local manifold learning-based *k*-nearest-neighbor for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4099–4109, Nov. 2010

Maadi, M., Javidnia, M., & Ghasemi, M. (2016). Applications of two new algorithms of cuckoo optimization (CO) and forest optimization (FO) for solving single row facility layout problem (SRFLP). *Journal of AI and Data Mining*, *4*(1), 35-48.

Marchiori, E. (2013, June). Class dependent feature weighting and k-nearest neighbor classification. In *IAPR International Conference on Pattern Recognition in Bioinformatics* (pp. 69-78). Springer, Berlin, Heidelberg.

Modha, D. S., & Spangler, W. S. (2003). Feature weighting in k-means clustering. *Machine learning*, *52*(3), 217-237.

Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Test functions for optimization needs*, 101.

Mazzola, S. (1990). A k-nearest neighbor-based method for the restoration of damaged images. *Pattern Recognition*, *23*(1-2), 179-184.

Paliwal, K. K., & Rao, P. V. S. (1983). Application of k-nearest-neighbor decision rule in vowel recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2), 229-231.

Paredes, R., & Vidal, E. (2006). Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(7), 1100-1110.

Sun, Y. (2007). Iterative RELIEF for feature weighting: algorithms, theories, and applications. *IEEE transactions on pattern analysis and machine intelligence*, *29*(6).

Tan, S., & Zhang, J. (2008). An empirical study of sentiment analysis for chinese documents. *Expert Systems with applications*, *34*(4), 2622-2629.

Todeschini, R. (1989). k-Nearest neighbour method: the influence of data transformations and metrics. *Chemometrics and intelligent laboratory systems*, *6*(3), 213-220.

Todeschini, R. (1990). Weighted k-nearest neighbour method for the calculation of missing values. *Chemometrics and Intelligent Laboratory Systems*, *9*(2), 201-205.

Triguero, I., García, S., & Herrera, F. (2010). IPADE: Iterative prototype adjustment for nearest neighbor classification. *IEEE Transactions on Neural Networks*, *21*(12), 1984-1990.

Triguero, I., Derrac, J., GarcíA, S., & Herrera, F. (2012). Integrating a differential evolution feature weighting scheme into prototype generation. *Neurocomputing*, *97*, 332-343.

Vinitski, S., Mohamed, F., Gonzalez, C., Knobler, R., Andrews, D., Iwanaga, T., & Madi, S. (1997). Fast tissue segmentation based on a 3D and 4D feature map: Comparison study. In *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE* (Vol. 2, pp. 505-508). IEEE.

Yue L., Tianlu Z., Wangwei J., & Siqi S. (2017). Materials discovery and design using machine learning. *Journal of Materiomics, 3, 1-19.*

Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, *40*(7), 2038-2048.

Wang, J., Neskovic, P., & Cooper, L. N. (2007). Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, *28*(2), 207-213.

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems* (pp. 1473-1480).

Wettschereck, D., & Aha, D. W. (1995, October). Weighting features. In *International Conference on Case-Based Reasoning* (pp. 347-358). Springer, Berlin, Heidelberg.

APPENDIX A

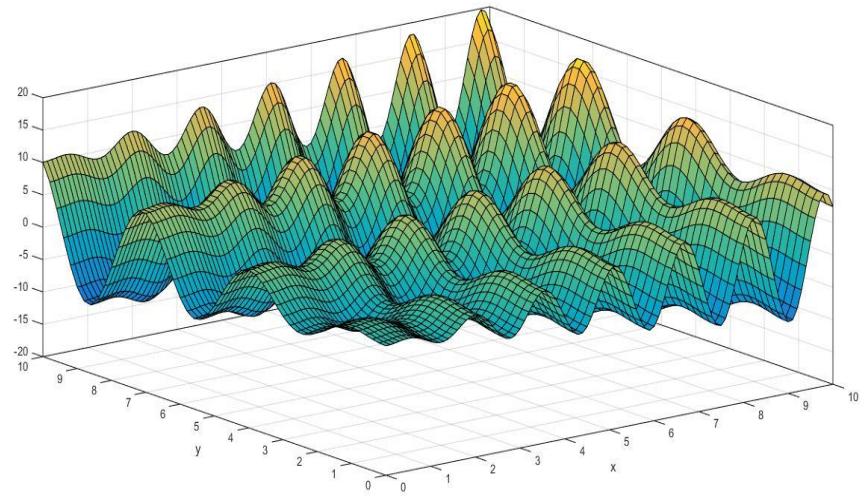

Figure 23: 3D graph of F1

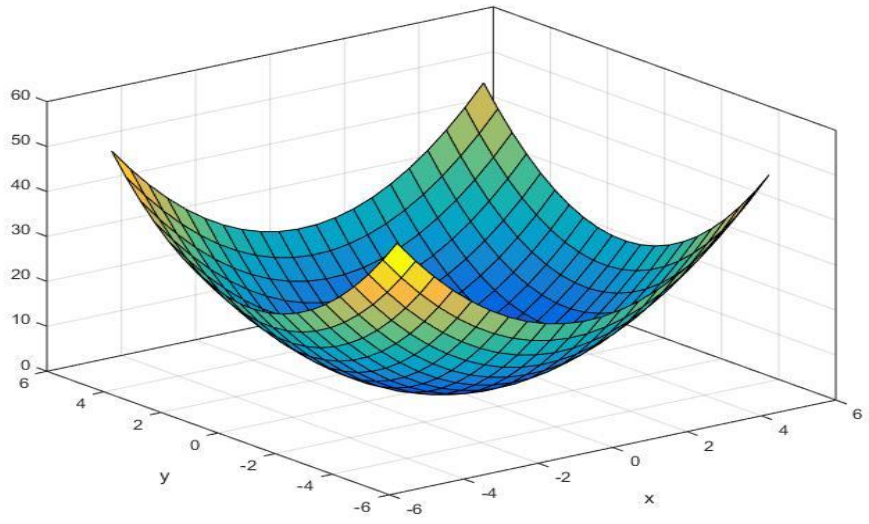

Figure 24: 3D graph of F2

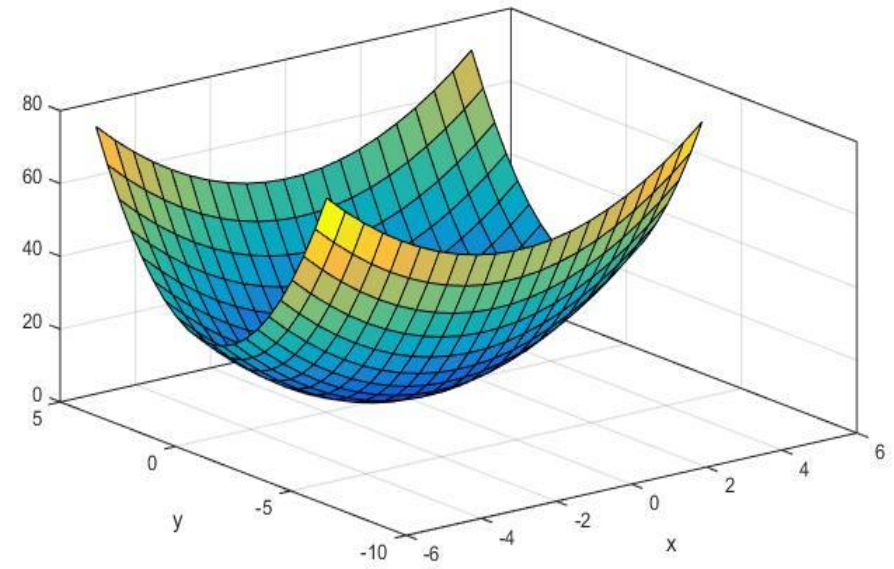

Figure 25: 3D graph of F3

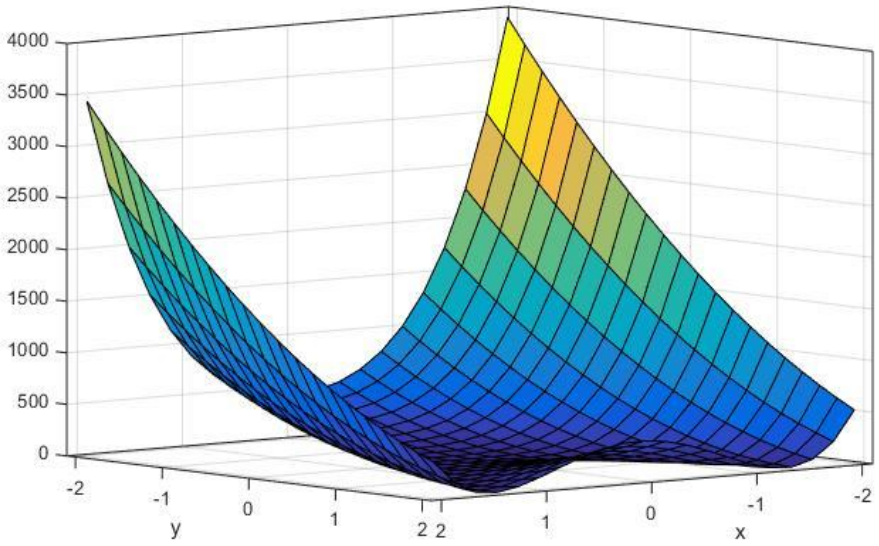

Figure 26: 3D graph of F4

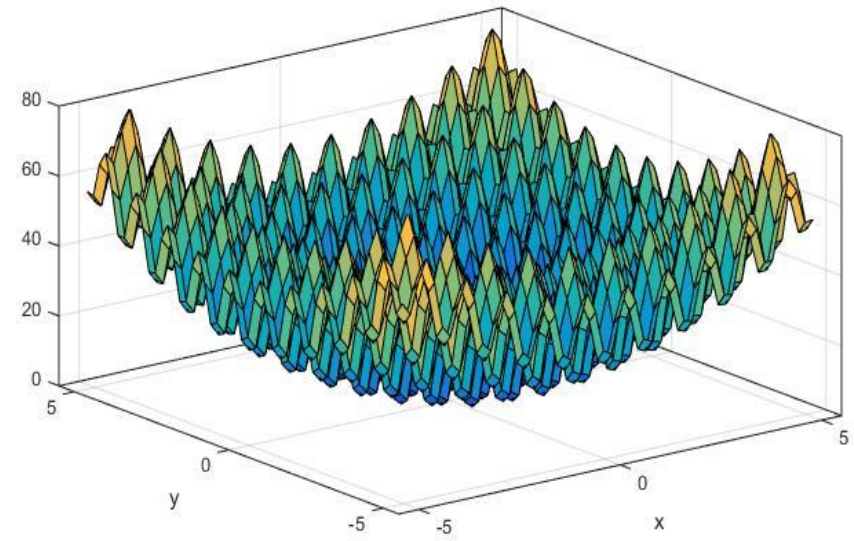

Figure 27: 3D graph of F5

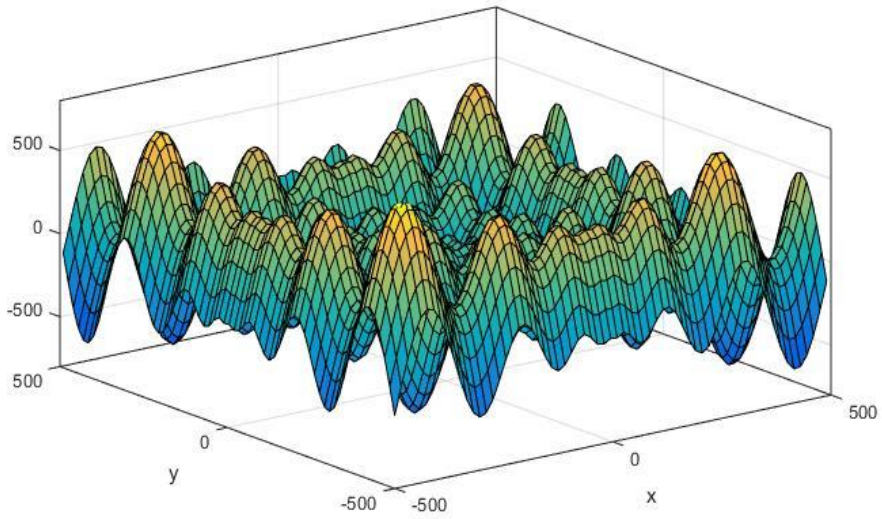

Figure 28: 3D graph of F6

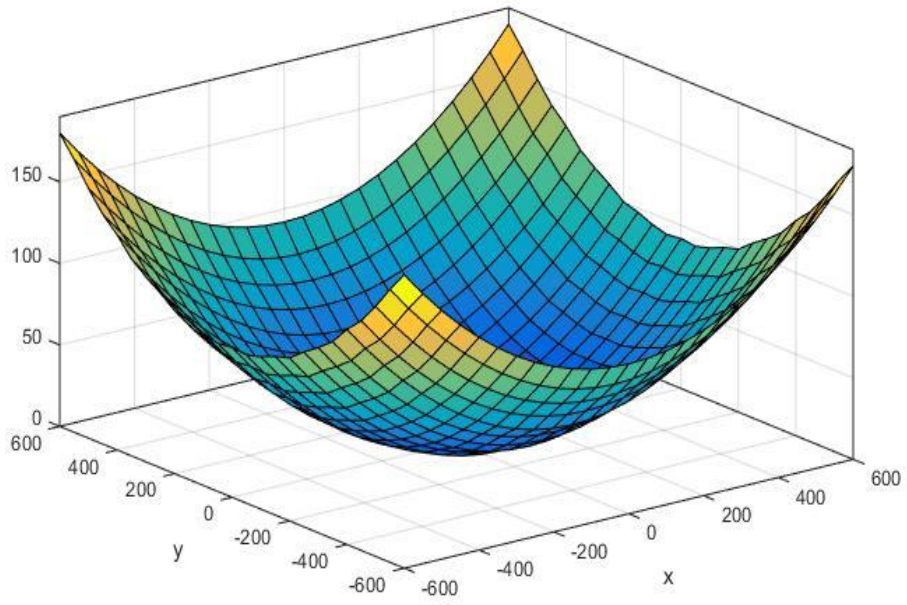

Figure 29: 3D graph of F7

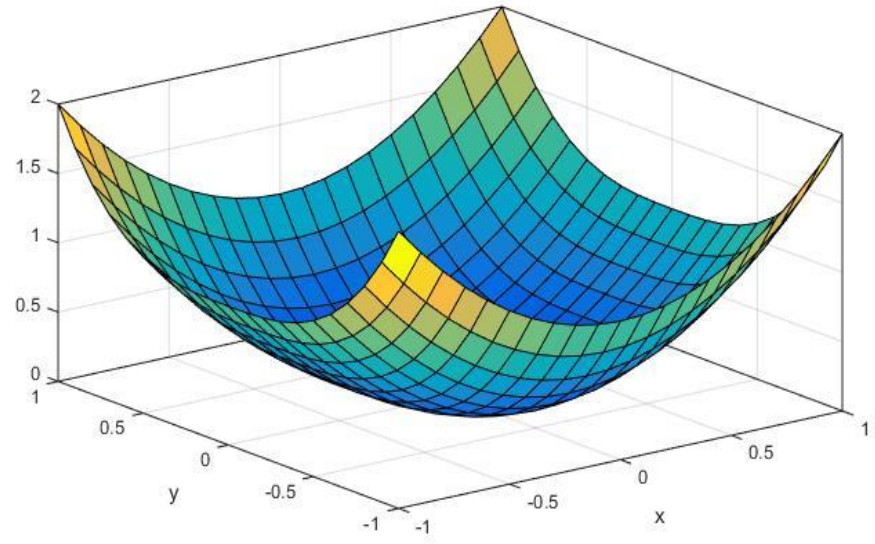

Figure 30: 3D graph of F8

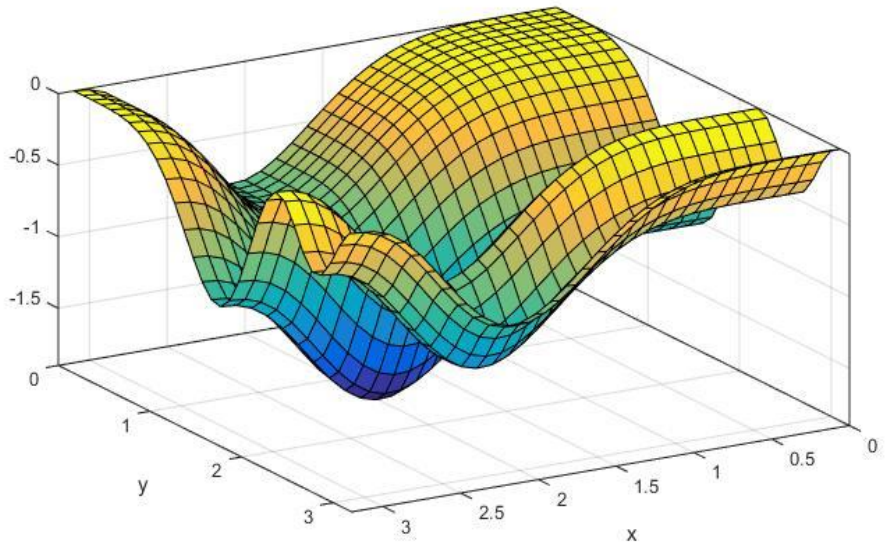

Figure 31: 3D graph of F9

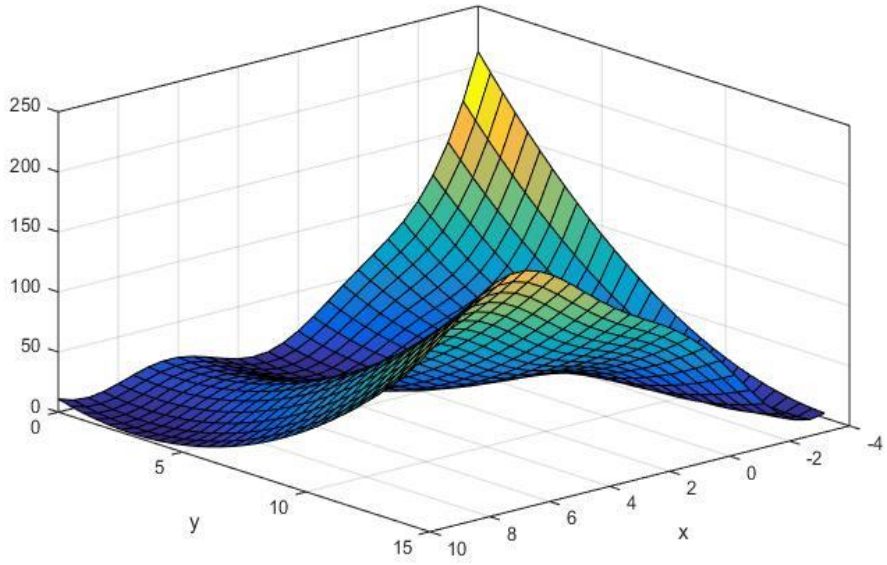

Figure 32: 3D graph of F10

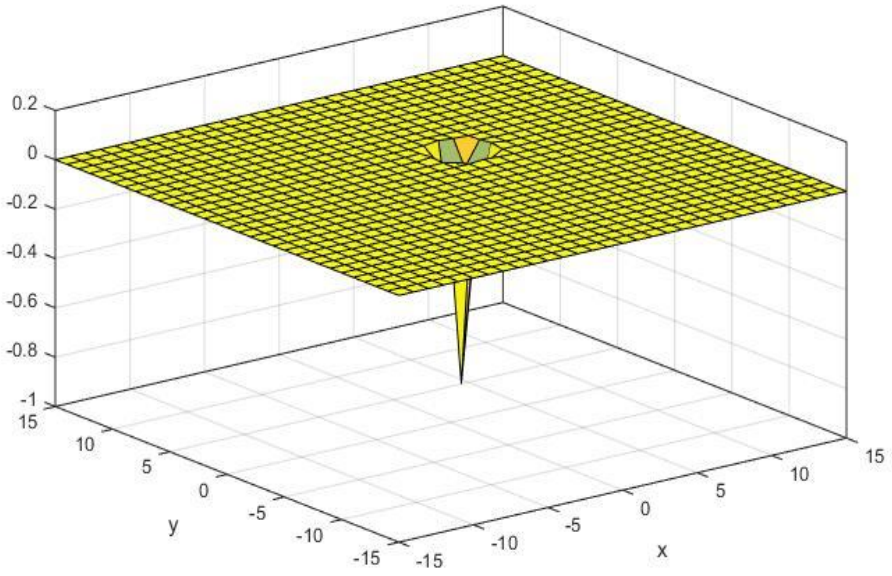

Figure 33: 3D graph of F11

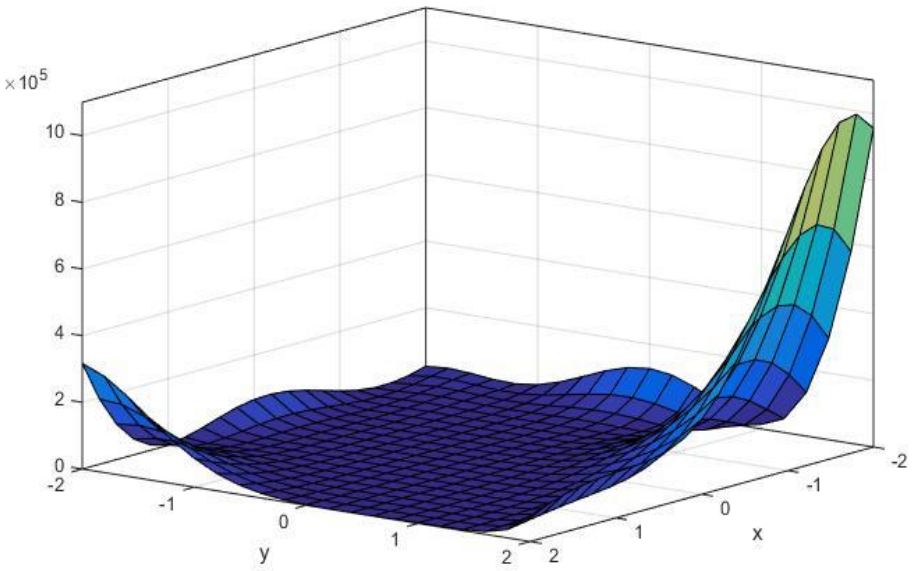

Figure 34: 3D graph of F12

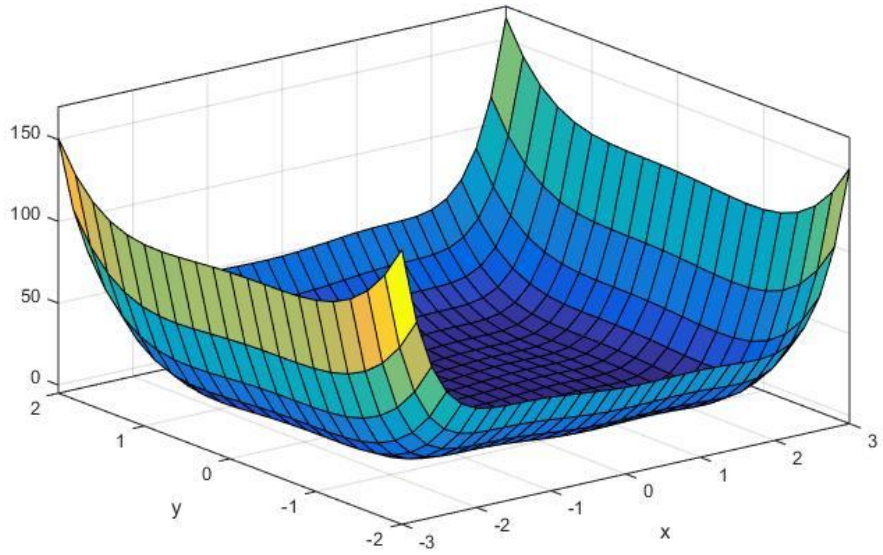

Figure 35: 3D graph of F13

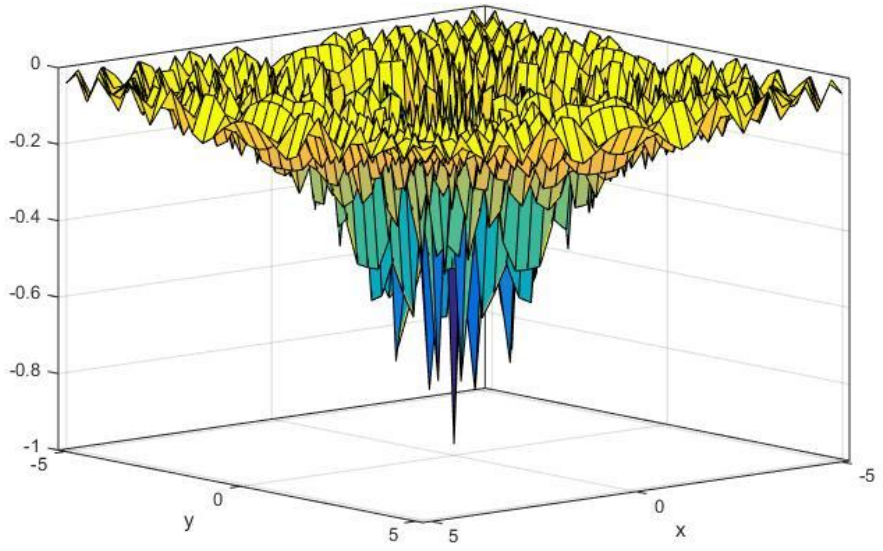

Figure 36: 3D graph of F14

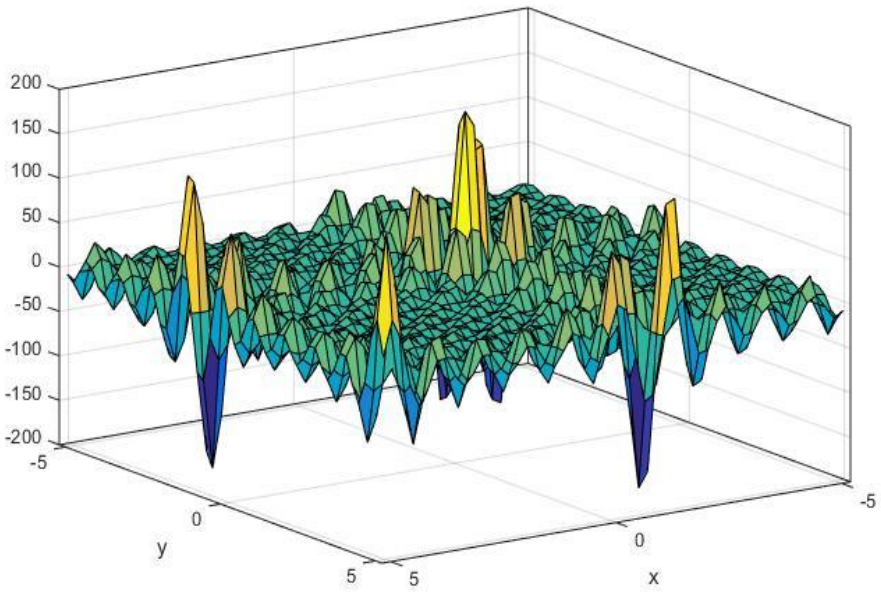

Figure 37: 3D graph of F15

APPENDIX B

The methods used here are Differential Evolution for Feature Weighting (DEFW) (Triguero, I., et al (2012)), Iterative Prototype Adjustment for Nearest Neighbor Classification (IPADECS) (Triguero, I., et al (2010)), hibritized version of IPADECS and DEFW method (IPADECS-DEFW) (Triguero, I., et al (2012)), Steady-state Memetic Algorithm (SSMA) (García, S. et al (2008)), hibritized version of SSMA and Differential Evolution for Prototype Generation and Feature Weighting (SSMA-DEPGFW), A Tabu Search Based Method for Simultaneous Feature Selection and Feature Weighting (TSKNN) (Triguero, I., et al (2012)).

All the methods in here are based on the 1NN rule. The purpose of the DEFW method is to find the most suitable weights using differential equations. The IPADE method attempts to determine the optimal number of prototypes for each class. This method consists of 3 basic levels; initialization, optimization, and addition of prototypes. The IPADECS-DEFW method is a hybrid method and aims to solve the feature weighting problem by combining the steps of two algorithms. In SSMA method, the aim is to find a solution to the evolutionary prototype selection problem using Memetic Algorithm. In SSMA-DEPGFW, another hybrid method, the processes of SSMA and DEPGFW are combined. In this method, the number of prototypes is determined first and then weights are calculated for the selected prototypes. In TSKNN method Feature Selection and Feature Weighting problems are discussed together. The Feature Selection Problem is solved first and then the Feature Weighting Problem is solved in order to find the appropriate weights for the selected features.