Attribute weighting in *K*-nearest neighbor classification

Muhammad Ejazuddin Syed

University of Tampere
School of Information Sciences
Computer Science
M.Sc. thesis
Supervisors: Kati Iltanen, Martti Juhola
November 2014

University of Tampere
School of Information Sciences
Computer Science
Muhammad Ejazuddin Syed: Attribute weighting in K-nearest neighbor classification
M.Sc. thesis, 50 pages and 2 index pages
November 2014

Abstract

Data mining is the process of getting useful information by analyzing different kind of data. Predictive data mining is used to predict some property of incoming data for example how to classify it. Among many methods that are used for predictive data mining the *K*-nearest neighbor classification is one of the simplest and easy to use technique. Due to its simplicity small variations are possible with it for the purpose of improving its predictive accuracy.

The aim of this thesis was to study attribute weighting techniques and to implement and test some weighting variants in *K*-nearest neighbor classification. The HEOM distance metric and three values of *K* (1, 4 and 5) were used in *K*-nearest neighbor classification. Twelve datasets were selected from the UCI Machine Learning Repository for the analysis. Chi-square attribute weighting was done in order to implement the two weighting variants. One variation was the simple attribute weighting and the other was the class-wise attribute weighting. The evaluation was done by using the leave-one-out technique.

The conclusion that can be drawn from the results is that the structure of the dataset (the number and the distribution of the classes) and the value of *K* (the number of neighbors) have effect on the unweighted and attribute weighted *K*-nearest neighbor classification. For some datasets weighting is very useful especially for smaller classes, but for some datasets it does not give improvements in the result.

Key words and terms: *K*-nearest Neighbor classification, attribute weighting

Contents

1. Introduction

Data mining means to extract useful knowledge from large amounts of data and it is one of the fast growing fields in *information technology*. Some people use data mining as a synonym to the term knowledge discovery from data (KDD), while others view data mining as an essential step in the KDD process which can be given as [Han and Kamber, 2006]:

- *Preprocessing:* Because knowledge discovery is often secondary usage of data (data have been originally collected for some other purposes), data have to be preprocessed (cleaned, integrated, transformed and reduced) before the actual data mining step.
- *Data mining:* The data are searched for to obtain interesting information or knowledge.
- *Postprocessing:* Discovered knowledge is evaluated and presented in an appropriate manner.

This thesis also uses the concept of machine learning which is a branch of *artificial intelligence* dealing with the construction of a machine or a system that learns from the data given to it. Learning here means that the system ends up a decision making pattern by analyzing and observing the data and results given to it and constantly updates this pattern whenever new data are presented to it. It is similar to the learning concept in human beings who make decisions based on their observations, that's why it is called artificial intelligence. Machine learning is used for many purposes but the main two are its common uses in pattern recognition and data mining.

There are two main purposes or tasks in data mining. In descriptive data mining different patterns and models are constructed that describe the data usually to find out associations or relations between data elements and their features. Finding values that occur frequently together in the given dataset is called as association analysis. One of descriptive data mining tasks is called cluster analysis which is partitioning data into groups so that the values in one group are similar to each other and are as different as possible from the values in other groups. Some application areas of association analysis are market basket analysis (which products are bought together), analysis of car crash data to find causes of accidents or identification of areas of similar land use in an earth observation database.

However this thesis is based on the second type of data mining called predictive data mining in which models that can be used to predict properties of unknown (new) data are constructed. Models describe and distinguish classes and give classes for new cases which is called classification. Some examples of classification are to give a diagnosis suggestion on the basis of the symptoms and test results of a patient and to predict the paying capacity of a loan applicant.

Classification is the task of assigning each record to one of the several predefined categories. Training set is the one whose class labels are known and along with the learning algorithm they are used to build a classification model which is applied to the test set whose class labels are unknown [Tan et al., 2006]. Each training data consist of a number of rows which are called records, tuples or instances. Input for classification is a set of records, where each record is characterized by values which are an attribute set and one is the special attribute designated as class label. A class label must always be discrete. Classification is also defined as the task of learning a target function that maps each attribute set to one of the predefined class labels. The target function is also called as classification model. Each classification technique employs a learning algorithm to identify a model that best fits the relationship between attribute set and class label of the input data. However there are some classification algorithms which do not make a model, but make the classification decision by comparing the test set with the training set each time they perform classification. These algorithms are known as instance-based learning algorithms.

There are many different classification algorithms present like decision tree induction, *K*-nearest neighbor classification, rule-based classifier, naïve Bayesian classifier, neural networks and support vector machines. This thesis concerns *K*-nearest neighbor classifiers which are instance-based learning algorithms.

As we have mentioned before, the data are not often collected originally for data mining purpose, so some attributes are irrelevant for classification and some attributes are redundant while other attributes are more important. Finding out the important attributes and giving them more impact on classification is called attribute weighting. One approach is just selecting important attributes and discarding the others while the other approach is giving all the attributes weights or ranks from the most important to the less important. In the thesis the second approach is used along with its two variants. One variant is that the single weights are calculated for every attribute from the whole training set and another variant is that different weights are calculated for different classes for every attribute, i.e., class-dependent weights are used. In the thesis weights were calculated by using the chi-square weight calculation for the Heterogeneous Euclidean Overlap Matric (HEOM) to find the nearest neighbor among the instances.

The main purpose of the thesis was to study attribute weighting in *K*-NN classification of medical datasets. For this purpose twelve datasets related to medical field specifically disease diagnosis were used. The aim was to compare both the attribute weighting variants to see which one gives better results.

The outline of the thesis is the following. Chapter 2 explains the *K*-nearest neighbor method and its variations in detail. Chapter 3 describes attribute weighting technique used in the *K*-nearest neighbor classification. Evaluation criteria and methods are explained in Chapter 4. Chapter 5 describes the experimental part of the thesis. Chapter 6 gives the results. Finally, discussion and conclusions are given in Chapter 7.

2. *K*-Nearest neighbor pattern classification

2.1 Concept

For classification of data it is extremely important that the nature of data is known which is either parametric or non-parametric. Parametric data means that there is some kind of statistical distribution between its instances which is known beforehand and vice versa, this information is essential because different algorithms and techniques are more suitable for different types of data. For parametric data there exist many different techniques, but Bayesian analysis will give the optimal result [Cover and Hart, 1967].

K-nearest neighbor classification is a classification technique that assumes the class of an instance to be the same as the class of the nearest instance. It adopts a similarity metric to measure the proximity of an instance to other instances. It is one of the most simple non-parametric decision rules. Here the term non-parametric refers to the fact that there is no prior knowledge of the statistical distribution of the data. Nearest neighbor assumes that instances in the data are independently and identically distributed, so the instances which are in close proximity have the same classification [Cover and Hart, 1967].

K-nearest neighbor classification is one type of instance-based learning methods which are sometimes called as lazy learning methods. "*K*-NN is purely lazy, it simply stores the entire training set and postpones all efforts towards inductive generalization until classification time"[Wettschereck et al., 1997] Instance-based learning methods are defined by their three properties:

- They store all of the training data during the learning process.
- Generalization beyond the training data is delayed until a value is predicted for a new case because any new query is answered by comparing the new case to the training data.
- From the training data they search for a case that is similar to the new case.

2.2 *K*-NN rule

In *K*-nearest neighbor classification, each instance is defined by a number of attributes and all the instances inside the data are represented by the same number of attributes, although there may be some missing attribute values. One of these attributes is called the class attribute which contains the class value (label) of the data, whose values are predicted for new, unseen instances.

1-NN rule assumes the value of the immediate neighbor to be the class of the new instance. However usually *K*-NN rule is used which assigns an instance the class which is represented mostly in its *K* neighbors. *K* can be any number of its neighbors, *K*= 1, 2, 3, 4,…,*n*, where *n* is the number of cases.

The closeness of a neighbor is defined on the basis of attributes defining the new instance and the training instances. Training instances whose attribute values are similar to that of the new instance are considered as the nearest, but many times the exact similar instance is not found, so the nearest instance is the one with least dissimilarity.

Assume:

m-dimensional attribute space.

C classes, numbered 1,2,...,*C*.

n training instances, each one expressed as a pair (x_i, θ_i), for $1< i <n$ where

a) x_i: training instance, expressed by a vector attribute va1ues.

$x_i = (x_{i1}, x_{i2,\ldots}, x_{im})$

b) $\theta_i \in \{1,2,...,C\}$ represents the correct class of the instance x_i.

Let $T_{NN} = \{(x_1,\theta_1), (x_2,\theta_2),\ldots, (x_n, \theta_n)\}$ be the nearest neighbor training set.

Given an unknown instance *x*, the decision rule is to decide *x* is in class

θ_j if $d(x,x_j) < d(x,x_i)$, for $1< i<n$, $i \neq j$

where *d* is some m-dimensional distance metric.

Figure 1. 1-NN formal definition [Vivencio et al., 2007].

Figure 1. describes specifically the 1-NN rule since it uses only one nearest neighbor in classification of a new instance, but it can be modified for the *K*-NN rule which considers *K* nearest instances $\{i_1, i_2,.., i_K\}$ and decides by assuming the most frequent class in the set $\{\theta_{i1}, \theta_{i2}, \ldots \theta_{iK}\}$.

In *K*-NN a small volume of the space of the attributes is taken and the new case is taken as the center of this volume. The radius of this volume is the distance from the new case to the K^{th} nearest neighbor. The estimated probability that this new case belongs to a certain class depends upon the relative frequencies of the classes of the training cases in this volume. The new case is assigned to the class that has the highest estimated probability [Hand et al., 2001].

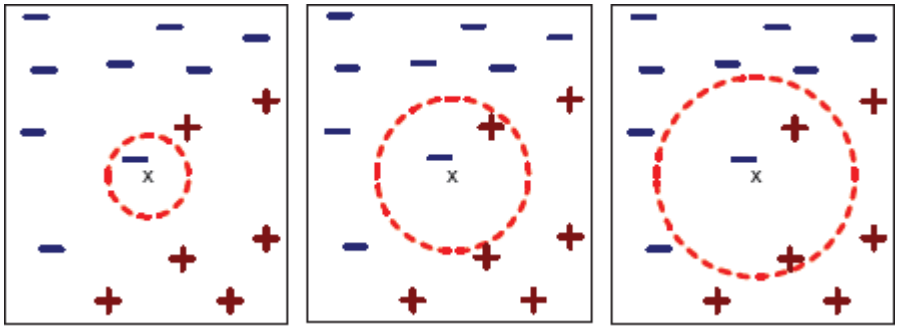

Figure 2. 1-NN, 2-NN and 3-NN. "+" and "-" are cases of positive and negative classes and "*x*" represents the new case. [Tan et al., 2006]

The basic form of K=1 in Figure 2(a) gives an unstable result because of its high variance and sensitiveness which makes it inconsistent to use [Hand et al., 2001], therefore larger value of K is used. The value of K that is to be used depends entirely upon the data set which is found out using different distribution analysis on the data. Figure 2(b) gives a tie situation in which two neighbors are selected which have different class values. This situation is resolved by probability sampling also called random selection. Figure 2(c) represents K=3.

It has been observed that if the training set is large enough K-NN yields good results. "In the large training set, this simple rule has a probability of error which is less than twice the Bayes probability of error, and hence is less than twice the probability of error of any other decision rule, nonparametric or otherwise, based on the infinite sample set" [Cover and Hart, 1967].

2.3 Proximity measures

As mentioned above nearest neighbor algorithm calculates the distance between the attributes of new instance and previous instances to find out the class. The term "distance" here depends entirely on the data, different types of data have different ways for finding out the distance. There are many different categorizations of attribute -data.

- Nominal attributes are categorically discrete data that consist of category names only and there is no distance between its values, i.e., they only differ in being similar or not (=, ≠). Binary attributes are a special case of nominal attributes.

- Ordinal attributes have a natural ordering to their values, but we cannot define the distance between them for example tall, medium and short are three attribute values that define a person's height but their difference cannot be given in numbers, i.e., we can only apply >,<,=,≠ to them.

- Interval attributes have some real number values and the difference between the values is meaningful. For example when measuring the temperature in Celsius there

is a meaningful difference in its values, i.e., there exist a distance between its values and >,<,=,≠,+,- operations can be performed on them.

- Ratio attributes are like interval attributes, but the difference is that in ratio attributes the value zero *"0"* is absolute zero, i.e., a characteristic disappears meaning the temperature in Kelvin scale or different kinds of counts and percentages are example of the ratio scale. Ratios of these values are meaningful.

Another categorization of attribute-data is between qualitative and quantitative.

- Qualitative attributes are categorical attributes usually expressed as category names by means of natural language. They can have order or no order between their values. Nominal and ordinal attribute-data are its two types depending on the ordering in its values.
- Quantitative attributes are expressed as numerical values. They describe the value as a measurable quantity, this value can be exactly measured in terms of numbers. However all numbers are not measurable like the social security number, therefore only measurable attributes are called quantitative attributes. Interval and ratio attribute-data are its two types.

There is another categorization of data.

- Homogeneous data: Data in which all the attributes are of same type. For example all of the attributes are nominal or interval type.
- Heterogeneous data: Data in which there are different types of attributes. For example one attribute is nominal while the other is interval.

2.3.1 Proximity measures for homogeneous data

Distance measures are defined by the proximity measures. Similarity measure means the more similar cases, the nearer they are to each other Dissimilarity measure means the more dissimilar cases, the farther they are from each other.

A metric is a dissimilarity function that fulfills four conditions [Hand et al., 2001]:

1. $d(x,y) \geq 0$ (for each *x* and *y* distances are nonnegative numbers)

2. $d(x,x) = 0$ (distance of an object to itself is zero) (also called reflexivity)

3. $d(x,y) = d(y,x)$ (symmetric)

4. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality: going directly from *x* to *y* is shorter (or equally short) than making a detour over object *z*.)

For a good learning system appropriate distance function is important. Wilson and Martinez [Wilson and Martinez, 1997] have listed a variety of distance functions. Here *x* and *y* are two input instances and *m* is the number of attributes.

- Euclidean distance function

$d(x,y) = \sqrt{\sum_{a=1}^{m}(x_a - y_a)^2}$

- City-block or Manhattan distance function

$d(x,y) = \sum_{a=1}^{m}|x_a - y_a|$

- Minkowskian r-distance function

$d(x,y) = \sqrt[r]{\sum_{a=1}^{m}(x_a - y_a)^r}$

- Mahalanobis distance function

$d(x,y) = [det\ V]^{1/m}\,(x - y)^T V^{-1}\,(x - y)$,
where V is a covariance matrix of A_1 to A_m

- Canberra distance function

$d(x,y) = \sum_{a=1}^{m}\frac{|x_a - y_a|}{|x_a + y_a|}$

- Chebychev distance function

$d(x,y) = max_{a=1}^{m}|x_a - y_a|$

- Quadratic distance function

$d(x,y) = (x - y)^T Q(x - y) = \sum_{b=1}^{m}\{\sum_{a=1}^{m}(x_a - y_a)q_{ab}\}(x_b - y_b)$,
where Q is a problem-specific positive definite m × m weight matrix

- correlation distance function

$d(x,y) = \frac{\sum_{a=1}^{m}(x_a - \overline{x}_a)(y_a - \overline{y}_a)}{\sqrt{\sum_{a=1}^{m}(x_a - \overline{x}_a)^2 \sum_{a=1}^{m}(y_a - \overline{y}_a)^2}}$

- chi-square distance function

$d(x,y) = \sum_{a=1}^{m}\frac{1}{sum_a}\left(\frac{x_a}{size_x} - \frac{y_a}{size_y}\right)^2$

where sum_a is the sum of all values for attribute *a* occuring in the training set and $size_x$ and $size_y$ are the sums of all values in the instance *x* and *y* respectively.

The above mentioned distance functions work well for quantitative attributes, but they do not have the solution for nominal, ordinal or heterogenuous data.

2.3.2 Proximity measures for heterogeneous data

Most of the real world applications including those that are present in UCI Machine learning repository have both qualitative and quantitative attributes. For heterogeneous datasets, distance functions are used that have the ability to accommodate both types of attributes.

Heterogeneous Euclidean-Overlap Metric (HEOM) was proposed by Wilson and Martinez [1997], It is a heterogeneous distance function that uses different distance functions on different types of attributes: overlap function is used for nominal attributes and normalized Euclidean distance for ordinal and quantitative attributes.

The distance between two input vectors *x* and *y* is given by the equation

$$\text{HEOM(x,y)} = \sqrt{\sum_{a=1}^{m} d(x_a, y_a)^2},$$

where *a* stands for an attribute and *m* stands for the total number of attributes

The distance between two values of input vectors *x* and *y* of a given attribute *a* is given by

$$d(x_a, y_a) = \begin{cases} 1, & (if\ x_a\ or\ y_a\ is\ unknown) \\ overlap\ (x_a, y_a), & (if\ a\ is\ nominal) \\ rn\_diff(x_a, y_a) & \end{cases}$$

If any value in the input vectors is unknown then for the attribute *a*, the maximum distance value is returned.

The overlap function is defined as

$$overlap(x_a, y_a) = \begin{cases} 0, (if\ x_a = y_a) \\ 1, (otherwise) \end{cases}$$

and the range normalized difference function is defined as

$$rn\_diff(x_a, y_a) = \frac{|x_a - y_a|}{range_a}$$

The range is used to normalize the attribute values and is defined as

$$range_a = max_a - min_a$$

where max_a and min_a are the maximum and minimum of attribute *a* taken from the training set.

As given above, the distance between two input vectors is the square root of the sum of the squared distances of all the attributes.

2.4 Improvements to *K*-NN

Many improvements or variations have been proposed in order to improve the performance and also reduce the shortcomings of the *K*-NN approach.

2.4.1 Density based *K*-NN classifier (DB-*K*NN)

Counting of the neighbors for determining the class of a test instance appears to be insufficient [Wang and Bell, 2004], due to the uneven distribution of training set, *K*-NN classification results will have relatively large differences. For the instances in dense area, its density is clearly higher than the instance in sparse area [Shi et al., 2011]. It can easily be understood that the similarity between instances in the areas that are densely populated is larger than the similarity between instances in the areas that are sparsely populated. *K* nearest neighbor classification depends upon the number of neighbors for the test instance and the density has an effect on this. The instances in the dense have more chance of selection than the instances in the sparse area. In short, if the decision function used does not consider the data distribution, then this will make the instance, the density of the category of which is dense, more likely to be selected. In the decision making process, it brings a negative impact on the classification results because the prediction of smaller classes is usually not accurate.

DB- KNN explores the potential of evaluating the neighbors in the *K*-NN rather than merely counting them [Voulgaris and Magoulas, 2008], they provides a new look to the *K*-NN algorithms. Also, the distance is used, making the evaluation of the neighbors more refined. In Density based *K*-NN the distance between test and training instances is increased in sparse area and reduced in dense areas because it not only considers the density of test instance but also the densities of its *K* neighbors.

2.4.2 Variable *K*-NN classifier (V-*K*NN)

From continuous experimentations it has been observed that the values in *K* nearest neighbor classification results heavily depends upon the number of neighbors (*K*) and each data has different *K* value that is suitable for it. It utilizes the concept of *degree of certainty* (DC) to do this.

Degree of certainty for one classification of a classifier [Voulgaris and Magoulas, 2008] is defined by the formula derived from the *certainty factor* CF formula in [Aydin and Guvenir, 2006].

$$DC_i = \frac{\max(classification\ score)}{\sum_{c=1}^{c'} classification\ score(c)}$$

where *i* denotes the *i-th* attribute classified, *c* is the class index, *c´* is the number of classes and *classification score* is the score determining the classification output of the classifier.

This approach finds the optimum *K* value for each classification [Voulgaris and Magoulas, 2008]. In this approach an array which contains the best *K* value for each training set instances is made. This array is called as "optimum" *K* array and is made by performing classification for each one of the training case instances based on various neighbors. The *K* value that gives the maximum degree of certainty of each classification is found. Therefore, for each training set there corresponds a particular *K* value which is considered the best and is stored in the "optimum" *K* array. Afterwards, for each test instance, the nearest neighbor is found and its *K* value is assumed based on the "optimum" *K* array. Then, the *K*-NN classifier is applied to that test instance, using that *K* value. Accoding to Voulgaris and Magoulas [2008] this is something similar to one of the ideas presented in [Khan et al., 2002].

2.4.3 Class based *K*-NN classifier (CB-*K*NN)

Many of datasets have the same common problem when it comes to classification that is the difference of their class sizes, meaning that one class will have too many instances while others (or some) have too few instances. When finding classification based upon the *K* nearest neighbor classification the classes with very few instances are not selected. The Class Based *K*NN (CB-*K*NN) was developed due to this fact [Voulgaris and Magoulas, 2008].

In this algorithm for every test instance, the *K* nearest instances of each class are taken. The value of *K* is automatically selected to maximize the degree of certainty of the classification, this selection is done by the classifier to decrease the influence of the most distance instance. Afterwards, the harmonic mean of the distances of these neighbors (the inverse of the mean of the inverses of distances) is calculated (so that it is not influenced so much by the most distant instances). Finally, these harmonic means are compared and the class yielding the lowest value is chosen for the classification.

2.4.4 Discernibility *K*-NN classifier (D-*K*NN)

This algorithm is similar in structure to the original *K*-NN extension of the DB-*K*NN. The aim was to make an algorithm that is quite fast, without losing accuracy [Voulgaris and Magoulas, 2008].

This algorithm takes into account the distance and discernibility of each neighbor. It takes the discernibility of each instance which is measured by taking a radius around each instance that is the average distance between this instance and the other instances of the same class. After that the instances with the same class and the total instances within that radius are divided and the result is called discernibility of that case. By dividing the discernibility by the distance, a score is produced, for each one of the

neighbors. Then, the scores for each class are averaged to produce one classification score for each one of the classes. The class yielding the highest classification score is selected for the classification.

2.4.5 Weighted *K*-NN classifier (W-*K*NN)

A weighted *K*-NN performs an evaluation on the attributes of the instances. This concept is similar to the Density Based *K*-NN classifier, which performs evaluation on the cases. Each attribute is evaluated to obtain a weight value based on how useful this attribute is for discerning the classes of the dataset. Just like in the Variable *K*-NN classifier these weights are stored in a vector format with one value for each attribute. The weighted *K*-NN classifier works in the manner that each one of the attributes of the training set are evaluated using the Index of Discernibility. The Index of Discernibility (ID) is a measure developed for assessing how easily distinguishable the classes of a dataset are and evaluating individual attributes by applying it on them. The weights are applied to both the training and the testing set when the distances between their cases are being calculated. This concept is further elaborated in Chapter 3.

3. Attribute selection and weighting

It has been frequently observed that the classification of data depends more upon some attributes than others. Finding those relevant attributes is called attribute selection and making a method in which those relevant attributes are given more value is called attribute weighting. Here the terms feature and attribute have the same meaning and are used as synonyms.

3.1 Attribute subset selection

The first task is to select the relevant features which is also called attribute subset selection. Attribute subset selection has become extremely important in areas where datasets with tens or hundreds of thousands of attributes are available [Guyon and Elissef, 2003]. On a simple thought having more attributes should only give us more benefits, but the real world provides us with many reasons why this is not generally the case [Koller and Sahami, 1996]. Reunanen [2003] observes that there can be many reasons for selecting only a subset of attributes:

(i) It is cheaper to measure only a subset of attributes;
(ii) By removing irrelevant attributes the prediction accuracy of the classification technique can be improved, i.e., to improve the prediction power of the method.
(iii) If less or few attributes are used, then the predictor which is used usually becomes simple and likewise its speed increases, i.e., the method becomes more cost effective by using less attributes.
(iv) It is useful to know the attributes which are relevant, as they can give insight into the nature of the prediction problem at hand.

Therefore, the problem of focusing on the most relevant information in a potentially overwhelming quantity of data has become increasingly important for machine learning and data mining procedures [Blum and Langley, 1997].

One question that comes to the mind is that what does the word “relevant” means. In machine learning literature, there exist different meanings of the word relevance. Blum and Langley [1997] have given some definitions regarding the word relevance in their paper. An attribute is relevant to the target concept if there exists any example in which changing the value of attribute affects the classification given by the target concept. In this thesis the target concept is the *K*-NN method. Similarly an attribute is irrelevant, if any change does not affect the predicted classification.

Attribute selection methods can be categorized into four basic steps that determine the nature of the heuristic search process. First, one must determine the starting point (or points) in the space, which in turn influences the direction of search and the operators used to generate successor states. A second decision involves the organization of the search. A third issue concerns the strategy used to evaluate alternative subsets of attributes. Finally, one must decide some criterion for halting the search.

Attribute selection methods can be grouped in two categories: "attribute weighting" and "minimum subset selection" algorithms [Liu and Motoda, 1998] based on the outputs of the method used. An attribute ranking algorithm defines a score to express the relevance of an attribute; a subset selection algorithm tries to identify a subset of relevant attributes. Liu and Motoda have explained “minimum subset selection” as deciding which attributes to use in describing the concept and deciding how to combine those attributes. In this view, it is the selection of relevant attributes, and the elimination of irrelevant ones.

3.2 Attribute subset selection approaches

3.2.1 Embedded approach

Embedded approaches perform attribute subset selection as a part of the learning procedure and are usually specific to given learning methods.

Greedy set cover method is a type of embedded method which is explained as follows: consider concepts that can be expressed as a disjunction of Boolean features. The concepts to be learned are given as classes positive and negative. To find minimum relevant attribute subsets begin with a disjunction of zero attributes (which by convention outputs “negative” on every example, “negative” is the default class). Then, out of those attributes not present in any negative example (“safe” attributes) choose the one whose inclusion into the current hypothesis most increases the number of correctly classified positive examples (breaking ties arbitrarily). Repeat this until there are no more “safe” attributes that would increase the number of correctly classified positives cases, and then halt. [Blum and Langley, 1997]

Quinlan’s ID3 [Quinlan 1983] and C4.5 [Quinlan 1993], and CART [Breiman et al., 1984] are some of the variations to this set cover method. These methods carry out a set cover search and at each stage use an evaluation function to select the attribute that has the best ability to discriminate among the classes. Embedded methods are less computationally intensive than other methods but are specific to learning methods.

3.2.2 Filter approach

Filter methods are used in the preprocessing phase, before the actual learning process. They attempt to assess the merits of attributes from the data, ignoring the effects of the selected attribute subset on the performance of the learning algorithm. They use general characteristics of the training set to select some attributes and exclude others. Thus, filtering methods are independent of the algorithm that will use their output, and they can be combined with any such algorithm.

FOCUS algorithm proposed by Almuallim and Dietterich [Almuallim and Dietterich, 1991] describe a filtering approach to attribute selection that involves a greater degree of search through the attribute space. The main function of this algorithm is to search

for minimum of attribute combinations that perfectly discriminate among the classes i.e. generate pure partitions of the training set in which no instances have different classes. This method first looks at single attribute, then combines two attributes, then three, and then four and finally it stops when it finds a perfectly discriminating combination. It then passes the original training examples that are described using only the selected attribute to the actual learning algorithm. This approach assumes that attributes are adequate for the classification tasks. Kira and Rendell's [1992] RELIEF algorithm uses ID3 to induce a decision tree from the training data using only the selected attributes. Kononenko [Kononenko, 1994] reports two extensions to this method that handle more general types of attributes.

Filter methods easily scale to very high-dimensional datasets, are computationally simple and fast, and independent of the classification algorithm. Attribute selection needs to be performed only once, and then different classifiers can be evaluated. They are often univariate or low-variate. This means that each attribute is considered separately, thereby ignoring attribute dependencies, which may lead to worse classification performance when compared to other types of attribute selection techniques.

3.2.3 Wrapper approach

Wrapper methods assess subsets of attributes according to their usefulness to a given predictor. The method conducts a search for a good subset using the learning algorithm itself as part of the evaluation function. The typical wrapper method searches for the same space of attribute subsets as embedded and filter methods, but it evaluates alternative sets by running some induction algorithm on the training data and using the estimated accuracy of the resulting classifier as its metric.

Langley and Sage's [Langley and Sage, 1994] OBLIVION algorithm combines the wrapper idea with the simple nearest-neighbor method, which assigns, to new instances, the class of the nearest case stored in memory during learning. The attribute -selection process effectively alters the distance metric used in these decisions, taking into account the attribute judged relevant and ignoring the others. This method carries out a backward elimination search through the space of attribute sets, starting with all attribute and iteratively removing the one that leads to the greatest improvement in estimated accuracy. The system continues this process until the estimated accuracy actually declines. Other examples are Race [Moore and Lee, 1994], Beam [Aha and Bankert, 1996] and Townsend-Weber & Kibler [Townsend-Weber and Kibler, 1994].

Wrapper method interacts between attribute subset search and model selection, and have the ability to take into account attribute dependencies, but they have higher risk of overfitting than filter techniques and are computationally very intensive, especially if building the classifier has a high computational cost.

3.3 Attribute weighting

Attribute weighting (as shortly described at the end of Chapter 2) assesses the importance (or relevance) of each attribute with respect to the output by using a univariate measure. Univariate means that the measurement is carried out for a single variable or value. Attribute weighting is mostly used because of its simplicity, scalability and good empirical success [Guyon and Elissef, 2003]. Variable ranking makes use of a scoring function that is computed from the attributes. The scoring function is produced in an increasing order. Examples are correlation technique and chi-square statistical test.

Some types of attributes ranking algorithms follow the idea of continuous weighting in which training instances lead to a simultaneous change in all attribute ranking values [Blum and Langley, 1997]. Examples are Perceptron updating rule [Minsky and Papert, 1969], which adds or subtracts weights on a linear threshold unit in response to errors on training instances, the least-mean squares algorithm [Widrow and Hoff, 1960], back propagation [Rumelhart et al., 1986] and Littlestone's [Limestone, 1988] WINNOW.

3.3.1 Pearson's correlation technique

Nowadays most commonly used method is called *Pearson's correlation* technique

$$R(i) = \frac{\sum_{k=1}^{n}(x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_{k,i} - \bar{x})^2 \sum_{k=1}^{n}(y_k - \bar{y})^2}}$$

To understand this equation consider a set of n instances $\{x_k, y_k\}$ (k = 1,..n) consisting of m input (known) $x_{k,i}$ (i = 1,..m) and one output(unknown, class) y_c attribute. Attribute ranking for Pearson's coefficient makes use of a scoring function $R(i)$ computed from the values $x_{k,i}$ and y_k, (k = 1,..m). Using this method as an attribute weighting criterion enforces a weighting according to goodness of linear fit of individual attribute [Guyon and Elissef, 2003]. Each attribute is judged individually and its value is calculated by computing with the class attribute. These values are saved in the form of a vector whose length is the total number of input attributes.

3.3.2 Chi-square test statistic

Another method for attribute weighting is called chi-square weighting which is used in the thesis given by [Liu and Motoda, 1998] and finds the value by calculating chi-square distance between each input attribute and class attribute.

The motivation for using chi-square as a mutual information measure in an attribute weighting task is due to the ability of this measure in ranking attributes [Witten and Frank, 2000]. Chi-square is designed to work for categorical data and it does not work

for quantitative data. It should also be known that the data should not be in the percentage form, only the frequency or count of data is needed.

One way of explaining the chi-square is that it checks the null hypothesis which states that there is no association between attributes. Based on this null hypothesis a model is created that distributed the data in categories assuming that there is no association between attributes. The test is based on comparing the actual distribution of the data and expected distribution of the data. A contingency table is needed to calculate chi-square.

Table1. Contingency table

Variable Y	Variable X				
	1	2	…	c	Totals
1	n_{11}	n_{12}	…	n_{1c}	$n_{1.}$
2	n_{21}	n_{22}	…	n_{2c}	$n_{2.}$
…	…	…	…	…	…
r	n_{r1}	n_{r2}	…	n_{rc}	$n_{r.}$
Totals	$n_{.1}$	$n_{.2}$	…	$n_{.c}$	n

The observed frequency for the cell c_{ij} is $n_{ij.}$ The expected frequency for the cell c_{ij} is e_{ij}

$$e_{ij} = \frac{n_{i.} n_{.j}}{n}$$

The chi-square test statistics is calculated as $(x^2) = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(n_{ij} - e_{ij})^2}{e_{ij}}$

From the equation we can see that the cells that contribute most to the chi-square value are the ones whose actual and expected counts are very different. The greater value represents that the attributes are related.

3.3.3 Sequential and normalized weighting

Although weight values can directly be used but there also exist many criteria for defining a weight vector. Mainly two criteria are used [Vivencio et al., 2007]. The first criterion, called Sequential Weighting (SW), defines a weight vector simply by ranking attribute, i.e., the attributes having the lowest score have their weights set to 1, those

with the second lowest-scored attributes have their weight set to 2 and so on. The process goes on until weights are assigned to the highest scored attributes. In datasets where all attributes have different scores, the highest scored attribute will have its weight set to number of attributes. The second criterion, called Normalized Weighting (NW), normalizes weights in the interval [0, 10]. According to this criterion, attributes with the highest score have their weights set to 10. The other attributes have their weights linearly established according to their score.

Attribute ranking method uses a vector that contains the weights or values corresponding to each attribute. These methods require the evaluation of each attribute using a specific distance metric, for identifying its degree of relevance (DR). The DR is then used to sort the attributes into a list called "ranked list of attribute" [Vivencio et al., 2007].

Attribute Ranking Algorithm

Input: dataset, distance measure;

Output: ranked list of attribute

List: ordered attribute list;

DR: degree of relevance function;

D_R: vector with the degree of relevance of each attribute

begin

List ← emptylist;

for each attribute f **do**

begin

D_R[f] ←

DR (*f*, dist_measure, dataset); /*degree of relevance */

List ← (*f*, *DR[f]*);

endfor;

order (List, *DR*);

end.

Figure 3. Attribute ranking algorithm [Vivencio et al., 2007].

3.3.4 Attribute weighted nearest neighbor

Attribute weighted nearest neighbor method can be classified as a mutual information approach for assigning attribute weights [Witten and Frank, 2005]. Therefore the chi-square statistical score between the values of attributes is used to assign the weights.

This method works in three steps. First, the chi-square score between each attribute and the class must be defined using the whole dataset; second, based on a weighting criterion, a vector containing the weights of each attribute is created and third, the weights of each attribute are used in the *K*-NN classification task. The following figure 4 shows the second and third step, first step has been shown in the figure 3 above.

Chi-Square attribute weighting algorithm

Input: dataset described by attributes plus class, criterion;

Output: Weight Vector (*V*)

Chi: vector with the *X* score for each attribute;

V: weight vector;

begin

for each attribute *f* **do**

Chi[f] ← *chi* (class, *f*, dataset);

V ← create_vector (criterion, *Chi*);

end.

Figure 4. Chi-Square attribute weighting algorithm [Vivencio et al., 2007].

4. Evaluation

4.1 Evaluation methods

There are many evaluation methods for classifiers presented over the years and in different perspectives, a classifier here means any method or algorithm that is used to classify the datasets. One way of differentiating evaluation methods is that by which factors the classifiers are being evaluated. In the objective evaluation performance measures are used to evaluate the performance of the classifiers, for example accuracy, true positive rate etc. In the subjective evaluation domain experts evaluate the classifier and the classifier should be intelligent and reasonable from their point of view.

Another two types of evaluation are descriptive and predictive [Hand et al., 2001]

- In the descriptive evaluation performance measures are calculated from the training instance to see how well the generated classifier labels the training instances, the training set is the one whose class labels are known and along with the learning algorithm are used to build a classification model. The disadvantage of this method is that if the same instance is used in the model construction and evaluation then the result will be overoptimistic.
- In the predictive evaluation the classifier training is done from the training set and the evaluation is done by a test instance or instances (on which the classifier has not been trained but whose class labels are known). Example methods are holdout method, cross validation method, etc.

4.1.1 Holdout method

One of the methods is called the holdout method. In this method the original data set is partitioned into two parts, the training set and test set [Han and Kamber, 2006]. The sets can be set as 50-50 or 2/3-1/3 for the training and test sets depending on the choice of the analyst. The data are randomly divided into the training and test sets, the classification model is induced from the training set and evaluated on the test sets. Random subsampling is a version in which the above method is repeated many times and a performance measure (e.g. the accuracy) is calculated as the average of the measures obtained in each iteration. Sampling means selecting a subset of individuals from the whole dataset. The holdout method is usually employed with large datasets, datasets having a large number of instances, probably around thousands of instances or more.

4.1.2 Cross-validation method

In the cross-validation each record is used for the same number of times for training but only once for testing. As an example two partitions of the data set are made and one partition is used for training and the other for testing and next time vice versa is done. This is called two-fold cross-validation. This can be generalized by partitioning the

dataset into *k* equal sized subsets. During each run one subset is used for testing while others are used for training. This process is repeated until each subset is tested once. This method is called *k*-fold cross-validation [Han and Kamber, 2006]. For example, we make k_1 to k_5 subsets.

- First make the subset k_2, k_3, k_4 and k_5 as the training set and k_1 as the testing set.
- Then use k_1, k_3, k_4 and k_5 as the training set and k_2 as the testing set.
- The process is repeated until all sets have become test sets one time.

Performance measures can be calculated in two ways either as the average of measures calculated in each iteration or on the basis of the overall number of correct classifications from all iterations. The cross-validation can be repeated with different divisions of subsets to acquire good knowledge about the dataset. Usually 10-fold cross validation is used as it has been tested and validated [Han and Kamber, 2006]. The cross-validation method is used with moderate datasets having instances around hundreds or more.

4.1.3 Leave-one-out validation

The leave-one-out validation is a special case of cross-validation method [Han and Kamber, 2006] in which each instance is used once as the test case and all other instances are used as the training set. If a dataset contains *n* instances then *n* -1 instances are used for training set and n^{th} instance for the test case. For this reason the leave-one-out method is also called as *n*-fold cross validation. This method utilizes the utmost training instances but due to its expensive nature it is usually applied to small datasets.

4.1.4 Bootstrap method

Another method for the evaluation is called bootstrap method [Han and Kamber, 2006]. This method samples the given training instances uniformly with replacement which means that each time an instance is selected it is equally likely to be selected again and added back to the training set. In sampling with replacement, the same instance is allowed to be selected more than once. There exist several bootstrap methods of which a commonly used one is the *.632 bootstrap*. In .632 method *n* is for the number of cases. The data set of *n* cases is sampled *n* times, with replacement, which results in a bootstrap sample or training set of *n* samples. In this sample some of the original instances will occur very likely more than once. The instances not occurring in the training set form the test set on average, 63.2% of the original instances will end up in the bootstrap, and the remaining 36.8% will form the test set.

4.2 Evaluation measures

4.2.1 Total accuracy and error rate

Accuracy is evaluated on the basis of records correctly or incorrectly predicted. A confusion matrix is constructed. A simple confusion matrix also called binary confusion matrix is given in Table2.

Table 2. Confusion matrix for 2 classes

		Predicted Class	
		Class = 0	Class = 1
Actual Class	Class = 0	f_{00}	f_{01}
	Class = 1	f_{10}	f_{11}

In the confusion matrix f_{00} means the number of records of the class 0 predicted in the class 0, f_{01} means the number of records of the class 0 predicted in the class 1, f_{10} means the number of records of the class 1 predicted in the class 0 and f_{11} means the number of records of the class 1 predicted in the class 1.

It is evident from the matrix that the correct predictions are $f_{00} + f_{11}$ and the incorrect predictions are $f_{01} + f_{10.}$ The accuracy is calculated as

accuracy = correct predictions / total predictions

$$= (f_{00} + f_{11}) / (f_{00} + f_{10} + f_{01} + f_{11}).$$

Similarly, the error rate is calculated as

error rate = incorrect predictions / total predictions

$$= (f_{10} + f_{10}) / (f_{00} + f_{10} + f_{01} + f_{11}).$$

This confusion matrix is for datasets having only two classification values. For datasets having more than two classification values this matrix can be modified, although the concept remains the same. A confusion matrix for a *C*-class classification tasks is given in table 3.

Table 3. Confusion matrix for n-classes

		Predicted Class			
		class 1	class 2	…	class *n*
Actual Class	class 1	f_{11}	f_{12}	…	f_{1C}
	class 2	f_{21}	f_{22}	…	f_{2C}
	…	…	…	…	…
	class *C*	f_{C1}	f_{C2}	…	f_{CC}

In the confusion matrix, f_{11} is the c number of cases whose actual class is class1 and the predicted class is class1 and f_{12} is the number of cases whose actual class is class1 and the predicted class is class2

The accuracy is calculated as the percentage

accuracy = $100\frac{(f_{11}+f_{22}+\cdots+f_{CC})}{n}$

where *n* is the total number of predictions

4.2.2 Class-wise accuracy

Another evaluation that is important is called class-wise accuracy, i.e., the proportion or percentage of correct predictions for instances of each class. It is important because it shows how much effect a particular class has on the overall accuracy and it simplifies the calculation of overall accuracy.

Let us assume that there are 100 cases present in a dataset which are described in the following way

- 94 class A instances
- 3 class B instances
- 3 class C instances

The total accuracy found out after calculation is 94% which is a very good result. However only the class A instances are correctly predicted and the other classes are not, which makes this result misleading.

For *class 1* the classwise accuracy is $100(f_{11}/f_{11}+f1_2+\ldots+f_{1C})$. This will give the percentage of instances correctly classified from *class 1*.

Generalizing,

$$ACC_c = 100 \frac{tn_c}{n_c},$$

where ACC_c is class-wise accuracy for the class c, tn_c is the number of correctly classified cases in the class c and n_c is the total number of cases in the class c.

From the above formula, the overall accuracy can be given by

$$ACC = 100 \frac{\sum_{c=1}^{C} tn_c}{\sum_{c=1}^{C} n_c}$$

where *c* is the class and *C* is the total number of classes.

5. Implementation of the experimental part

5.1 Datasets

Twelve different datasets from the UCI Machine learning repository were selected for the calculations. The UCI Machine Learning Repository is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms [UCI, 2014]. The default task for the datasets was classification and they belonged to the medical field. Details of the datasets are as follows.

Audiology

- 221 cases
- 19 classes: class attribute – "Diseases"
  Class "1" – "cochlear_age" (57 cases),
  Class "2" – "cochlear_age_and_noise" (22 cases),
  Class "3" – "cochlear_noise_and_heredity" (2 cases),
  Class "4" – "cochlear_poss_noise" (20 cases),
  Class "5" – "cochlear_unknown" (48 cases),
  Class "6" – "conductive_discontinuity" (2 cases),
  Class "7" – "conductive_fixation" (6 cases),
  Class "8" – "mixed_cochlear_age_fixation" (2 cases),
  Class "9" – "mixed_cochlear_age_otitis_media" (4 cases),
  Class "10" – "mixed_cochlear_age_s_om" (2 cases),
  Class "11" – "mixed_cochlear_unk_discontinuity" (2 cases),
  Class "12" – "mixed_cochlear_unk_fixation" (9 cases),
  Class "13" – "mixed_cochlear_unk_ser_om" (3 cases),
  Class "14" – "mixed_poss_noise_om" (2 cases),
  Class "15" – "normal_ear" (22 cases),
  Class "16" – "otitis_media" (4 cases),
  Class "17" – "possible_brainstem_disorder" (4 cases),
  Class "18" – "possible_menieres" (8 cases),
  Class "19" – "retrocochlear_unknown" (2 cases)
- 70 attributes
- 62 nominal and 8 ordinal attributes

Balance Scale

- 625 cases
- 3 classes : class attribute – "Balance Scale"
  Class "B" – "Balanced" (49 cases),
  Class "L" – "Tip to left" (288 cases),
  Class "R" – "Tip to Right" (288 cases)
- 4 attributes
- 4 ordinal attributes

Breast cancer

- 286 cases
- 2 classes:
  Class “1” – “no-recurrence-events” (201 cases),
  Class “2” – “recurrence-events” (85 cases)
- 9 attributes
- 5 nominal, 2 interval and 2 ratio attributes

Bupa

- 345 cases
- 2 classes: class attribute – “Liver Disorder from excessive alcohol consumption”
  Class “1” – “Yes” (145 cases),
  Class “2” – “No” (200 cases)
- 6 attributes
- 6 nominal attributes.

Contraceptive method choice

- 1473 cases
- 3 classes : class attribute – “contraceptive method used”
  Class “1” – “not used” (629 cases),
  Class “2” – “long term” (333 cases),
  Class “3” – “short term” (511 cases)
- 9 attributes
- 5 ordinal and 4 interval attributes

Dermatology

- 366 cases
- 6 classes: class attribute – “Eryhemato-Squamous Disease”
  Class “1” – “psoriasis” (32 cases),
  Class “2” – “seboreic dermatitis” (123 cases),
  Class “3” – “lichen planus” (32 cases),
  Class “4” – “pityriasis rosea” (123 cases),
  Class “5” – “cronic dermatitis” (32 cases),
  Class “6” – “pityriasis rubra pilaris” (123 cases)
- 34 attributes
- 33 ordinal and 1 interval attributes

Heart disease

- 303 case
- 5 classes: class attribute – "angiographic disease status"
  "Diameter narrowing" means blockade/narrowing in the blood vessels in diameters that effects the blood flow.
  Class "0" – "< 50% diameter narrowing" (164 cases),
  Class "1" – "< 60% & > 50% diameter narrowing" (55 cases),
  Class "2" – "< 70% & > 60% diameter narrowing" (36 cases),
  Class "3" – "< 80% & > 70% diameter narrowing" (35 cases),
  Class "4" – "> 80% diameter narrowing" (13 cases)
- 75 attributes of which only 14 are used
- Original attributes are of different types but they have been converted into ordinal form.

Hepatitis

- 155 cases
- 2 classes: class attribute – "status"
  Class "1" – "live" (32 cases),
  Class "2" – "die" (123 cases)
- 19 attributes
- 13 nominal , 5 ordinal and 1 interval attributes

Indian liver patient

- 583 cases
- 2 classes: class attribute – "liver patients"
  Class "1" – "yes" (416 cases),
  Class "2" – "No" (167 cases)
- 10 attributes
- 1 nominal, 7 ordinal, 1 interval and 1 ratio attributes

Pima Indian Diabetes

- 768 cases
- 2 classes: class attribute – "Diabetes"
  Class "0" – "negative" (500 cases),
  Class "1" – "positive" (268 cases)
- 8 attributes
- 7 interval and 1 ratio attributes

Post-Operative

- 90 cases
- 3 classes: class attribute – "Discharge decision"
  Class "I" – "patient sent to Intensive Care Unit" (2 cases),

Class “S” – “patient prepared to go home” (24 cases),
Class “A” – “patient sent to general hospital floor” (364 cases),

- 8 attributes
- 7 ordinal and 1 ratio attributes

Spectf

- 267 cases
- 2 classes: class attribute – “Spectf Heart Disease”
  Class “0” – “negative” (55 cases),
  Class “1” – “positive” (212 cases)
- 44 attributes
- 44 ratio attributes

5.2 *K*-NN impementation

The UCI Machine Learning Repository datasets were given as CSV (comma separated values) text files. CSV means that the attribute values in the text file were separated by a comma “,”. Another text file was also given which contains the details about the datasets, for example the number of attributes, values of all the attributes (for calculating range), types of attribute values (nominal, ordinal etc.), missing values (how they are indicated by), the class attribute etc.

Three versions of the *K*-NN program were made. In the first version, attribute weighting was not applied, but the basic *K*-nearest neighbor classification was done. In the second version, a simple weight addition was applied where the weights were calculated from the training instances for every attribute. In the third version, the class-wise weight addition was applied in which different weights were calculated for each class in a class versus others manner for every attribute from the training instances.

The data file was converted into a two-dimensional array where each instance was represented by a row and each attribute of that instance was represented by a column of that row. For the calculation purpose the values of the whole dataset were converted into a numeric form. The string values were mapped into a discrete numeric form like 1, 2,…,etc. and the class attribute was saved in another array and was removed from the dataset because the predicted class was to be found out by using the *K*-nearest neighbor algorithm.

The leave-one-out validation technique was applied to the *K*-NN variants, one row was considered as a test instance and all others as training instances. This was repeated for each row. The distance between the test instance and each instance in the training set was calculated by using the Heterogeneous Euclidean Overlap Metric (HEOM) distance measure. For the second and third versions of *K*-NN weights based on the chi-square statistic were multiplied by the distance measure to give the new distance value.

The chi-square weights were calculated for each attribute from the whole training set and these values were saved in a weight array. Since the chi-square test is only applicable for qualitative values, discretization was done to convert quantitative values into qualitative values. Class-wise weights were implemented by calculating the chi-square weights from the training instances in a class versus the other classes manner for every attribute, so the number of weight vector was equal to the number of classes and the weight array in this case was two-dimensional. The weight value that would be multiplied by the distance measure would be determined by the class the training instance belonged to.

The nearest instance to the test instance was the one which has the lowest distance value. Since *K*-NN involves taking more than one nearest neighbor, different numbers of neighbors were taken: K=1 where only the nearest one was taken, K=4 where four neighbors were taken in order to observe the behavior of even number of neighbors and K=5 taking five neighbors.

The class values of the neighboring instances were observed and the class that appeared the most times was taken as the predicted class of the test case. For K=1 the single neighbors class was taken as the predicted value. In the case of a tie between the class values appearing most a random selection was done between those values. This process was repeated for all the instances in the dataset, the predicted class values of all the instances were calculated.

By using a confusion matrix, the overall accuracy and class-wise accuracies were calculated.

6. Results

6.1 Individual dataset results

The classification results for the Audiology data are given in Table 5. It can be observed that for all values of *K* (1, 4 and 5), the best results for all the classes were distributed between all the versions with version 2(weighted *K*-NN) having the majority of the best results, and also the best results for overall accuracy for all values of K (73.7% – 81.9%). version 1 also has best results for many classes.

The classification result for the balance scale are given in Table 6. It can be seen that version 1 (unweighted *K*-NN) produces the best results for the two majority classes (54.9% - 97.9%) and total accuracy (55.7% – 71.4%), for all value of *K* (1,4 and 5) with the exception of one majority class value when *K*=1(44.4) which comes from version 2 (weighted *K*-NN). *K*=5 produced the best total accuracy result (71.4%) and it can be seen that total accuracy increases with increasing value of *K*. Another interesting fact for this dataset is that version 3 (class-wise *K*-NN) produces the best result for the minority class for all values of *K* (100%) while other two versions were unable to correctly classify this class. But version 3 was unable to classify the other two classes at all (0%).

The classification results for the Breast Cancer data are given in Table 7. It can be observed that the version 1 (basic or unweighted *K*-NN) produces the best results (89.5, 91.1 and 95.5%) for the majority class (class "1") with all the values of *K* (1, 4 and 5 respectively), but the weighted versions 2 and 3 produce better results (44.7 and 28.2) for the minority class (class "2"). Since the dataset contains only 2 classes, the results for the weighted versions 2 and 3 are usually the same but there can be small differences because of the involvement of the random selection in the tie situations. For *K*=4, the probability to have tie situations is greater than for *K*=1 or *K*=5, and accordingly there is a greater difference in the results of 4-NN.

The classification results for the Bupa data are given in Table 8. From these results it can be observed that version 1 (unweighted *K*-NN) produces best results for the majority class (68.5% - 69.5%) for all values of *K* (1, 4 and 5) and for the minority class for *K*=4 and *K*=5, and the best total accuracy for all values of *K*.

The classification results for the Contraceptive Method Choice data are given in Table 9. It can be seen that the version 3 (class-wise weighted *K*-NN) produces the best results for the minority class (class "2") (39.3 – 43.8 %) and also for class "3" (47.9 – 61.1%) for all the values *K* (*K*=1, 4 and 5). The Total accuracy is best for the version 1 (unweighted *K*-NN) (47.0%). This is due to the majority class (class"1") which constitute the biggest part in the total accuracy (50.4 – 52.3%), *K*=4 produces the best result for the total accuracy. *K*=4 produces the best results for the majority class (class"1") and also for the minority class (class"2").

The classification results for the Dermatology data are given in Table 10. It can be observed that version 2 (weighted *K*-NN) produces the best results (96.4% - 99.1%) for

the majority class (class "1") for all values of *K* (1, 4 and 5) and also for many of the other classes ("2", "4","5" and "6") and the best overall accuracy (97%) when *K*=5, Version 1 (unweighted *K*-NN) produces the best overall accuracy (92.9% - 95.6%) for *K*=1 and *K*=4. Version 3 (class-wise *K*-NN) produces the best results for the third biggest class (class "2") for *K*=1 and *K*=4.

The classification results for the Heart Disease data are given in Table 11. It has been observed that the version 2 (weighted *K*-NN) produces the best results in most cases for all the classes. Version 2 produces the best results for minority class (class "4") (7.7 – 23.1%) for all values of *K* and majority class for *K*=1 (82.3%). The Version 2 also produces the best total accuracy for *K*=1 (56.4%) and *K*=4 (59.1%) while the version 1 (unweighted *K*-NN) produces the best total accuracy for *K*=5 (59.7%). Version 1 gives best results for the majority class (class"0") for *K*=4 (88.4%) and *K*=5 (92.1%). *K*=5 gives best total accuracy and best result for majority class while *K*=1 gives best minority class (class"4"). A unique pattern in these results is of the second biggest class (class"1") whose results in version 3 (class-wise weighted) (56.6 – 72.7%) are by far greater than the versions 1 (30.9 – 34.5%) and 2 (12.7 – 29.1%) for all values of *K*.

The classification results for the Hepatitis data are given in Table 12. It can be observed that the version 1 (unweighted *K*-NN) produces the best results for the majority class (class "2") with *K*=4 (93.5%) and *K*=5 (94.3%) but for *K*=1 version 2 (weighted *K*-NN) produces the best accuracy (90.2%). Version 1 also produces the best results for the minority class (class "1") and the total accuracy for all values of *K*. *K*=1 gives the best result for minority class (50%) while *K*=5 gives best result for the majority class (class "2") (94.3%) and the total accuracy (83.9%).

The classification results for the Indian Liver Patient data are given in Table 13. It can be observed that the version 3 (class-wise *K*-NN) produces the best results for the majority class (class "1") (75.2 – 79.6%), the minority class (class "2") (28.7 – 53.3%) and the total accuracy (65.0 – 69%) for all values of *K*, although version 2 (weighted *K*-NN) also gives the best results for *K*=1 and *K*=5 because there are only 2 classes but for *K*=4 version 3 gives the best results due to tie situations. *K*=5 gives the best result for the majority class (class "1") while *K*=1 gives best result for minority class (class "2") and the total accuracy.

The classification results for the Pima Indian Diabetes data are given in Table 14. It can be seen that version 2 (weighted *K*-NN) and version 3 (class-wise *K*-NN) produces best results for the minority class (class "1") for all values of *K* (1, 4 and 5) and produced the best total accuracy (75.4%) when *K*=5. Since, there are two classes, the results of version 2 and version 3 are the same. Version 1 (unweighted *K*-NN) produces the best results for the majority class (class "0") and total accuracy for *K*=1 and *K*=4.

The classification results for the Post-Operative Patient data are given in Table 15. It can be observed that version 2 (weighted *K*-NN) produces the best results (75% - 90.6%) for the majority class (class "A") and best total accuracy (57.8% - 68.9%) for all values of *K* (1, 4 and 5). Version 3 (class-wise *K*-NN) produces the best results (33.3% - 50%) for the second smallest class (class "S").

The classification results for the Spectf data are given in Table 16. It can be observed that version 2 (weighted *K*-NN) produces the best results for majority class (class "1") (79.2% - 85.4%), for minority class (class "0") (38.2% - 43.6%) and total accuracy (70.8% - 76%) for all values of *K* (1, 4 and 5). Version 3 (class-wise weighting) shares the same results for *K*=1 and *K*=5.

Table 5. The K-NN classification results for the Audiology data from five times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	57	53	93	55	**96.5**	52	91.2
2	22	16	72.7	21	**95.4**	12	54.6
3	2	2	**100**	1	50	1	50
4	20	11	55	13	65	17	**85**
5	48	38	79.2	40	83.3	44	**91.7**
6	2	2	**100**	2	**100**	1	50
7	6	6	**100**	5	83.3	5	83.3
8	2	0	0	0	0	1	**50**
9	4	0	0	1	**25**	1	**25**
10	2	0	0	0	0	0	0
11	2	1	50	2	**100**	1	50
12	9	8	88.9	9	**100**	5	55.5
13	3	3	**100**	3	**100**	2	66.7
14	2	2	**100**	2	**100**	2	**100**
15	22	21	**95.4**	18	81.8	12	54.6
16	4	0	0	0	0	0	0
17	4	4	**100**	4	**100**	2	50

18	8	4	50	5	**62.5**	2	25
19	2	0	0	0	0	0	0
Total Accuracy		77.4		**81.9**		72.4	
K=4							
1	57	55	**96.5**	53	93	53	92.3
2	22	13	59.1	16	**72.7**	14	63.6
3	2	0	0	0	0	0	0
4	20	9	45	11	55	14	**70**
5	48	40	83.3	43	**89.5**	42	87.5
6	2	0	0	0	0	1	**50**
7	6	6	**100**	3	50	5	83.3
8	2	0	0	0	0	0	0
9	4	0	0	3	**75**	1	25
10	2	0	0	0	0	0	0
11	2	0	0	0	0	0	0
12	9	8	88.9	9	**100**	5	55.6
13	3	2	**66.7**	2	**66.7**	0	0
14	2	0	0	1	50	0	0
15	22	20	**90.9**	16	72.7	12	54.6
16	4	0	0	0	0	1	**25**
17	4	1	25	2	**50**	0	0
18	8	1	12.5	3	**37.5**	0	0

19	2	0	0	0	0	0	0
Total Accuracy		70.1		**74.7**		67	
K=5							
1	57	54	**94.7**	54	94.7	51	89.5
2	22	12	54.5	17	**77.3**	15	68.2
3	2	0	0	0	0	0	0
4	20	10	50	10	50	14	**70**
5	48	36	75	42	87.5	45	**93.8**
6	2	0	0	0	0	0	0
7	6	6	**100**	5	83.3	5	83.3
8	2	0	0	0	0	0	0
9	4	0	0	2	**50**	1	25
10	2	0	0	0	0	0	0
11	2	0	0	0	0	0	0
12	9	7	77.8	9	**100**	5	55.6
13	3	3	**100**	3	**100**	0	0
14	2	0	0	0	0	0	0
15	22	19	**86.4**	17	77.3	11	50
16	4	0	0	0	0	0	0
17	4	1	25	2	**50**	0	0
18	8	1	**12.5**	1	**12.5**	0	0
19	2	0	0	0	0	0	0
Total Accuracy		67.4		**73.3**		66.5	

Table 6. The K-NN classification results for the Balance scale data from 5 times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
B	49	0	0	0	0	49	**100**
L	288	68	23.6	128	**44.4**	0	0
R	288	280	**97.2**	124	43.2	0	0
Total Accuracy		**55.7**		40.3		7.8	
K=4							
B	49	0	0	0	0	49	**100**
L	288	158	**54.9**	127	44.1	0	0
R	288	277	**96.2**	128	44.4	0	0
Total Accuracy		**69.6**		40.80		7.8	
K=5							
B	49	0	0	0	0	49	**100**
L	288	164	**56.9**	121	42	0	0
R	288	282	**97.9**	118	41	0	0
Total Accuracy		**71.4**		38.2		7.8	

Table 7. The *K*-NN classification results for the Breast Cancer data from five times repeated leave-one-out.

		Version 1 (Basic *K*-NN)		**Version 2 (Weighted *K*-NN)**		**Version 3 (Class-wise weighted *K*-NN)**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	201	180	**89.5**	164	81.6	163	81.1
2	85	26	30.6	38	**44.7**	38	**44.7**
Total Accuracy		**72.0**		70.6		70.3	
K=4							
1	201	183	**91.0**	163	81.1	162	80.6
2	85	23	27.1	24	**28.2**	22	25.9
Total Accuracy		**72.0**		66.4		64.3	
K=5							
1	201	192	**95.5**	166	82.6	165	82.1
2	85	20	**23.5**	19	22.3	19	22.4
Total Accuracy		**74.1**		64.7		64.3	

Table 8. The K-NN classification results for the Bupa data from 5 times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	145	78	53.8	80	**55.2**	80	**55.2**
2	200	139	**69.5**	136	68	136	68
Total Accuracy		**62.9**		62.6		62.6	
K=4							
1	145	78	**53.8**	76	52.4	76	52.4
2	200	137	**68.5**	127	63.5	127	63.5
Total Accuracy		**62.3**		58.8		58.8	
K=5							
1	145	72	**49.7**	71	49	71	49
2	200	139	**69.5**	135	67.5	135	67.5
Total Accuracy		**61.2**		59.7		59.7	

Table 9. The *K*-NN classification results for the Contraceptive Method Choice data from 5 times repeated leave-one-out.

		Version 1 (Basic *K*-NN)		**Version 2 (Weighted *K*-NN)**		**Version 3 (Class-wise weighted *K*-NN)**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	629	323	**51.4**	309	49.1	213	33.9
2	333	117	35.1	119	35.7	131	**39.3**
3	511	210	41.1	209	40.9	245	**47.9**
Total Accuracy		**44.1**		43.2		40	
K=4							
1	629	329	**52.3**	301	47.8	159	25.3
2	333	133	39.9	130	39.0	146	**43.8**
3	511	231	45.2	222	43.4	305	**59.7**
Total Accuracy		**47.0**		44.3		41.4	
K=5							
1	629	317	**50.4**	306	48.6	141	22.4
2	333	129	38.7	118	35.4	135	**40.5**
3	511	241	47.2	234	45.8	312	**61.1**
Total Accuracy		**46.6**		44.7		39.9	

Table 10. The K-NN classification results for the Dermatology data from 5 times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	112	108	**96.4**	108	**96.4**	16	14.3
2	61	48	78.7	51	**83.6**	51	**83.6**
3	72	71	**98.6**	66	91.7	14	19.4
4	49	44	89.8	46	**93.9**	43	87.8
5	52	51	98.1	52	**100**	43	82.7
6	20	18	**90**	16	80	13	65
Total Accuracy		**92.9**		92.6		49.2	
K=4							
1	112	109	97.3	110	**98.2**	5	4.5
2	61	53	86.9	54	88.5	58	**95.1**
3	72	72	**100**	71	98.6	9	12.5
4	49	46	**93.9**	42	85.7	40	81.6
5	52	50	96.2	52	**100**	43	82.7
6	20	20	**100**	20	**100**	7	35
Total Accuracy		**95.6**		95.4		44.3	
K=5							
1	112	109	97.3	111	**99.1**	3	2.7
2	61	55	90.2	56	**91.8**	55	90.2
3	72	72	**100**	71	98.6	6	8.3
4	49	49	**100**	45	91.8	42	85.7
5	52	50	96.2	52	**100**	41	78.8
6	20	18	90	20	**100**	7	35
Total Accuracy		96.4		**97**		42.1	

Table 11. The *K*-NN classification results for the Heart Disease data from 5 times repeated leave-one-out.

		Version 1 (Basic *K*-NN)		**Version 2 (Weighted *K*-NN)**		**Version 3 (Class-wise weighted *K*-NN)**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
0	164	131	79.9	135	**82.3**	82	50
1	55	19	34.6	16	29.1	31	**56.6**
2	36	8	22.2	10	**27.8**	7	19.4
3	35	6	17.1	7	**20**	6	17.1
4	13	3	**23.1**	3	**23.1**	3	**23.1**
Total Accuracy		55.1		**56.4**		42.6	
K=4							
0	164	145	**88.4**	143	87.2	64	39.0
1	55	17	30.9	15	27.3	36	**65.4**
2	36	3	8.3	8	**22.2**	5	13.9
3	35	8	22.9	10	**28.6**	3	8.6
4	13	1	7.7	3	**23.1**	1	7.7
Total Accuracy		57.4		**59.1**		36	
K=5							
0	164	151	**92.1**	143	87.2	60	36.6
1	55	18	30.9	7	12.7	40	**72.7**
2	36	4	11.1	11	**30.6**	3	8.3
3	35	7	**20**	6	17.1	2	5.7
4	13	1	**7.7**	1	**7.7**	0	0
Total Accuracy		**59.7**		56.4		34.6	

Table 12. The *K*-NN classification results for the Hepatitis data from 5 times repeated leave-one-out.

		Version 1 (Basic *K*-NN)		**Version 2 (Weighted *K*-NN)**		**Version 3 (Class-wise weighted *K*-NN)**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	32	16	**50**	14	43.8	14	43.8
2	123	109	88.6	111	**90.2**	111	**90.2**
Total Accuracy		**80.6**		**80.6**		**80.6**	
K=4							
1	32	15	**46.9**	13	40.6	12	37.5
2	123	115	**93.5**	114	92.7	115	**93.5**
Total Accuracy		**83.9**		81.9		81.9	
K=5							
1	32	14	**43.8**	10	31.2	10	31.2
2	123	116	**94.3**	115	93.5	115	93.5
Total Accuracy		**83.9**		80.6		80.6	

Table 13. The *K*-NN classification results for the Indian Liver Patient data from 5 times repeated leave-one-out.

		Version 1 (Basic *K*-NN)		**Version 2 (Weighted *K*-NN)**		**Version 3 (Class-wise weighted *K*-NN)**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
1	416	300	72.1	313	**75.2**	313	**75.2**
2	167	78	46.7	89	**53.3**	89	**53.3**
Total Accuracy		64.8		**69**		**69**	
K=4							
1	416	318	76.4	314	75.5	321	**77.2**
2	167	58	34.7	55	32.9	59	**35.3**
Total Accuracy		64.5		63.3		**65.2**	
K=5							
1	416	328	78.8	331	**79.6**	331	**79.6**
2	167	44	26.4	48	**28.7**	48	**28.7**
Total Accuracy		63.8		**65.0**		**65.0**	

Table 14. The K-NN classification results for the Pima Indian Diabetes data from 5 times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
0	500	398	**79.6**	391	78.2	391	78.2
1	268	141	52.6	147	**54.8**	147	**54.8**
Total Accuracy		**70.2**		70.0		70.0	
K=4							
0	500	411	**82.2**	403	80.6	403	80.6
1	268	138	51.5	143	**53.4**	143	**53.4**
Total Accuracy		**71.5**		71.1		71.1	
K=5							
0	500	416	83.2	424	**84.8**	424	**84.8**
1	268	149	55.6	155	**57.8**	155	**57.8**
Total Accuracy		73.6		**75.4**		**75.4**	

Table 15. The K-NN classification results for the Post-Operative Patient data from 5 times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
A	64	45	70.3	48	**75**	40	62.5
I	2	0	0	0	0	0	0
S	24	1	4.2	4	16.7	8	**33.3**
Total Accuracy		51.1		**57.8**		53.3	
K=4							
A	64	55	85.9	58	**90.6**	41	64.1
I	2	0	0	0	0	0	0
S	24	1	4.2	4	16.7	12	**50**
Total Accuracy		62.2		**68.9**		58.9	
K=5							
A	64	57	89.1	58	**90.6**	38	59.4
I	2	0	0	0	0	0	0
S	24	0	0	2	8.3	9	**37.5**
Total Accuracy		63.3		**66.7**		52.2	

Table 16. The K-NN classification results for the Spectf data from 5 times repeated leave-one-out.

		Version 1 Basic K-NN		**Version 2 Weighted K-NN**		**Version 3 Class-wise K-NN**	
Class Name	**Total Number of Cases**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**	**Correctly Predicted Cases**	**Class Accuracy (%)**
K=1							
0	55	20	36.4	21	**38.2**	21	**38.2**
1	212	153	72.2	168	**79.2**	168	**79.2**
Total Accuracy		64.8		**70.8**		**70.8**	
K=4							
0	55	21	38.2	24	**43.6**	20	36.4
1	212	170	80.2	171	**80.7**	170	80.2
Total Accuracy		71.5		**73**		71.2	
K=5							
0	55	12	21.8	55	**40**	55	**40**
1	212	180	84.9	212	**85.4**	212	**85.4**
Total Accuracy		71.9		**76**		**76**	

6.2 Combined Result

For comparison all the results of datasets have been combined into Table 17. In table 17 accuracies and median true positive rates (TPR) are given. Median TPR, also called as sensitivity, is calculated by means of the class-wise accuracies, it is the median of the accuracies of the classes. It can be observed that out of 12 datasets the version 1 (Basic *K*-NN) has the best total accuracy and median TPR for 6 of them (Balance scale, Breast Cancer, Bupa, Contraceptive Method Choice, Dermatology, Hepatitis) for all values of *K* except for *K*=1 for Breast Cancer data, *K*=1 for Balance scale and *K*=5 Dermatology. The version 2 (weighted *K*-NN) has best accuracy and median TPR for audiology (*K*=4), Heart Disease (*K*=1 and *K*=4), Indian Liver Patient (with all *K*-values), Pima Indian Diabetes (*K*=5), Indian Liver Patient dataset and Pima Indian dataset have only two classes, so the differences between version 2 and 3 are because of random choices in the tie situations. Version 2 also has best accuracy and median TPR for spectf (with all *K*-values), the Post-Operative data for all values of *K* and the best median TPR for Dermatology (*K*=4 and *K*=5). Heart Disease for *K*=1 and *K*=4, for Audiology *K*=1 and *K*=5 version 1 had best accuracies, for Post-Operative version 3 has best median TPR for all values of *K*. The version 3 (Class-wise weighted *K*-NN) has the best median TPR for Post-operative for all values of *K*. It can also be observed that the value of accuracy increases with *K*=5 in 8 (Balance Scale, Breast Cancer, Contraceptive Method Choice, Dermatology, Heart Disease, Hepatitis, Pima Indian, Spectf) out of 12 datasets.

Table 17. Combined *K*-NN classification results of 12 dataset with total accuracy and median true positive rate (TPR)

DATASET	*K*	Version 1 (Basic *K*-NN)		Version 2 (weighted *K*-NN)		Version 3 (Class-wise weighted *K*-NN)	
		Accuracy	True positive rate	Accuracy	TPR	Accuracy	TPR
Audiology	1	**77.4**	79.2	70.6	**83.3**	72.4	50
	4	70.1	12.5	**74.7**	**50**	67	25
	5	**67.4**	12.5	64.7	**50**	66.5	0
Balance scale	1	**55.7**	23.6	40.3	**43.2**	7.8	0
	4	**69.6**	**54.9**	40.8	44.1	7.8	0
	5	**71.4**	**56.9**	38.2	42	7.8	0
Breast Cancer	1	**72.0**	60.1	70.6	**63.1**	70.3	62.9
	4	**72.0**	**59.0**	66.4	54.7	64.3	53.2
	5	**74.1**	**59.5**	64.7	52.5	64.3	52.2
Bupa	1	**62.9**	**61.6**	62.6	**61.6**	62.6	**61.6**
	4	**62.3**	**61.1**	58.8	58	58.8	58
	5	**61.2**	**59.6**	59.7	58.2	59.7	58.2
Contraceptive Method Choice	1	**44.1**	**41.1**	43.2	40.9	40	39.3
	4	**47.0**	**45.2**	44.3	43.4	41.4	43.8
	5	**46.6**	**47.2**	44.7	45.8	39.9	40.5
Dermatology	1	**92.9**	**93.2**	92.6	92.8	49.2	73.85

	4	**95.6**	96.8	95.4	**98.4**	44.3	58.3
	5	96.4	96.8	**97**	**98.8**	42.1	56.9
Heart Disease	1	55.1	23.1	**56.4**	**27.8**	42.6	23.1
	4	57.4	22.9	**59.1**	**27.3**	36	13.9
	5	**59.7**	**20**	56.4	17.1	34.6	8.3
Hepatitis	1	**80.6**	**69.3**	**80.6**	67	**80.6**	67
	4	**83.9**	**70.2**	81.9	66.5	81.9	65.5
	5	**83.9**	**69.0**	80.6	62.4	80.6	62.4
Indian Liver Patient	1	64.8	59.4	**69**	**64.3**	**69**	**64.3**
	4	64.5	55.6	63.3	54.2	**65.2**	**56.2**
	5	63.8	52.6	**65.0**	**54.2**	**65.0**	**54.2**
Pima Indian Diabetes	1	**70.2**	66.1	70	**66.5**	70	**66.5**
	4	**71.5**	66.8	71.1	**67**	71.1	**67**
	5	73.6	69.4	**75.4**	**71.3**	**75.4**	**71.3**
Post-Operative	1	51.1	4.2	**57.8**	16.7	53.3	**33.3**
	4	62.2	4.2	**68.9**	16.7	58.9	**50**
	5	63.3	0	**66.7**	8.3	52.2	**37.5**
Spectf	1	64.8	54.3	**70.8**	**58.7**	**70.8**	**58.7**
	4	71.5	59.2	**73**	**62.2**	71.2	58.3
	5	71.9	53.4	**76**	**62.7**	**76**	**62.7**

7. Discussion and conclusion

The main purpose of this thesis was to study the effects of attribute weighting on *K*-nearest neighbor classification. In addition to the attribute weighting, there are many factors within the *K*-nearest neighbor algorithm, for example the number of neighbors *K* and the distance measure used to find nearest neighbors. In the thesis, three values of *K* (1,4,5) were considered and the Heterogeneous Euclidean-Overlap Metric (HEOM) was used. The reason for using this distance measure was the fact that the HEOM method is used to calculate the distance of both qualitative and quantitative attributes. When doing attribute weighting in the *K* nearest neighbor classification, the weighting technique naturally plays a significant role in classification results as well as the approach used to implement this weighting technique. In the thesis the chi-square weighting technique was implemented by using the chi-square test statistics. Two variations of variable weighting were implemented, one was the normal weighting in which a single weight was calculated for every attribute while the other was the class-wise weighting in which different weights were calculated for different classes for every attribute.

After the evaluation of the results there are some points that can be derived from them. It was seen that there was a significant difference in their results. Usually *K*=5 gave the best results but not in all cases. Also weighting had advantageous results for many datasets. Similarly class-wise weighting was also very useful particularly in recognizing the smaller classes and improving the rate of accuracy for these smaller classes. Although the total accuracy in many cases was greater in the unweighted (without weights) method due to the fact that the bigger classes are easily predicted in the unweighted method, the weighting is, however useful for smaller classes and so possibly improves the median true positive rate.

The limitations of the thesis were that only twelve datasets were used evaluate the attribute weighting effect on the *K*-nearest neighbor algorithm. Out of these twelve, six datasets had only two classes, so the effects of class-wise attribute weighting cannot be generalized. More tests are needed with other datasets.

The future work could be to find the reasons why class-wise attribute weighting could not correctly classify the majority class but it correctly recognizes the minority class in some datasets, also to alter properties of the *K*-nearest neighbor algorithm which have been given above. The classification can be done by taking more and higher values of *K* so that the most suitable value of *K* could be found out which may increase the correct classification rate or by the finding best *K* using the variable *K*-NN method as explained in Chapter 2. Similarly, the distance measure could be changed to some distance measure other than HEOM which can calculate the distance for both the qualitative and quantitative attributes. Lastly, some other attribute weighting technique or other attribute selection approach could be used.

REFERENCES:

[Aha and Bankert, 1996] D.W. Aha and R.L. Bankert, A comparative evaluation of sequential feature selection algorithms, In: D. Fisher and J.-H. Lenz (eds.*), Arttficial Intelligence and Statistics V,* Springer, New York, 1996.

[Almuallim and Dietterich, 1991] H. Almuallim and T.G. Dietterich, Learning with many irrelevant features, In: *Proc. Association for the Advancement of Artificial Intelligence (AAAI-91),* 547-552.

[Aydin and Guvenir, 2006] T. Aydin and H. A. Guvenir, Modeling Interestingness of Streaming Classification Rules as a Classification Problem. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, ISSN 1611- 349, Volume 3949, 2006

[Blum and Langley, 1997] A. L. Blum and P. Langley, Selection of relevant features and examples in machine learning. *Artificial Intelligence.* **97**(1-2), 1997, 245-271.

[Breiman et al., 1984] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and Regression Trees*. Wadsworth Belmont, 1984.

[Cover and Hart, 1967] T. Cover and P. Hart, Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **13**(1), 1967, 21-27.

[Guyon and Elissef, 2003] I. Guyon and A. Elisseeff, An introduction to variable and feature selection. *Journal of Machine Learning Research*. **3,** 2003, 1157-1182.

[Han and Kamber, 2006] J. Han and M. Kamber, *Data Mining Concepts and Techniques*. Morgan Kaufmann, 2006.

[Hand et al., 2001] D. Hand, H. Mannila and P. Smyth, *Principles of Data Mining.* MIT Press, 2001.

[Shi et al., 2011] K. Shi, L. Li, H. Liu, J. He, N. Zhang, W. Song, An improved KNN text classification algorithm based on density. In: *Proc. of IEEE Cloud Computing and Intelligence Systems,* 2011, 113 – 117

[Khan et al., 2002] M. Khan, Q. Ding and W. Perrizo, K-Nearest Neighbors Classification of Spatial Data Streams using P-trees. In: *Proc of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD),* 2002, 517-528

[Kira and Rendell, 1992] K. Kira and L. Rendell, A practical approach to feature selection, In: *Proc. 9th international Conference on Machine Learning*, 249-256.

[Koller and Sahami, 1996] D. Koller and M. Sahami, Toward optimal feature selection. In: *Proc. of the 13th International Conference on Machine Learning.* 1996, 284-292.

[Kononenko, 1994] I. Kononenko, Estimating attributes: analysis and extensions of RELIEF, In: *Proc. 7th European Conference on Machine Learning*, 1994, 171-182.

[Langley and Sage, 1994] P. Langley and S. Sage, Oblivious decision trees and abstract cases, In: *Proc. Working Notes af the AAAI-94 Workshop on Case-Based Reasoning.* 1994, 113-l17.

[Limestone, 1988] N. Limestone, Learning quickly when irrelevant attributes abound: a new linear threshold algorithm, *Machine Learning* **(2)**, 1988, 285-318.

[Liu and Motoda, 1998] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic, 1998.

[Minsky and Papert, 1969] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, 1969.

[Mitchell, 1997] T. M. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[Moore and Lee, 1994] A.W. Moore and M.S. Lee, Efficient algorithms for minimizing cross validation error, In: *Proc. 11th International Conference on Machine Learning*, New Brunswick, 1994, 190-198.

[Moreno-Seco et al., 2003] F. Moreno-Seco, L. Mico and J. Oncina, A Modification of the LAESA Algorithm for Approximated k-NN Classification. *Pattern Recognition Letters.* **24**, 2003, 47–53.

[Quinlan, 1993] J.R. Quinlan, C4.5: *Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Quinlan, 1983] J.R. Quinlan, Learning efficient classification procedures and their application to chess end games, In: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (eds), *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, 1983.

[Reunanen, 2003] J. Reunanen, Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research.* **3**, 2003, 1371-1382.

[Rumelhart et al., 1986] D.E. Rumelhart, G. Hinton and R.J. Williams, Learning internal representations by error propagation, In: D.E. Rumelhart and J.L. McClelland, and the PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* **(1)**, MIT Press, Cambridge, 1986.

[Tan et al., 2006] P.N. Tan, M. Steinbach, V. Kumar, *Introduction to Data Mining*, Addison Wesley, 2006, http://www-users.cs.umn.edu/~kumar/dmbook/index.php (Checked on June 25th, 2014)

[Townsend-Weber and Kibler, 1994] T. Townsend-Weber and D. Kibler, Instance-based prediction of continuous values, In: *Proc. Working Notes af the AAAI-94 Workshop on Case-Based Reasoning.* 1994, 30-35.

[UCI, 2007] UCI Machine Learning Repository http://archive.ics.uci.edu/ml/about.html (Checked on April 15th, 2014)

[Vivencio et al., 2007] D. P. Vivencio, E. R. Hruschka, M. do Carmo Nicoletti, E. B. dos Santos, S. D. C. O. Galvio, Feature-weighted k-nearest neighbor classifier. In: *Proc. IEEE Symposium on Foundations of Computational Intelligence.* 2007, 481-486.

[Voulgaris and Magoulas, 2008] Z. Voulgaris and G. D. Magoulas, Extensions of the k Nearest Neighbour Methods for Classification Problems. In: *Proc of the 26th IASTED International Conference on Artificial Intelligence and Applications,*23-28, 2008

[Wang and Bell, 2004] H. Wang and D. Bell, Extended k-Nearest Neighbours Based on Evidence Theory. *The Computer Journal*, **47**(6), 2004, 662- 672.

[Wettschereck et al., 1997] D. Wettschereck, D. W. Aha, T. Mohri, A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* **11(**1-5**)**, 1997, 273-314.

[Widrow and Hoff, 1960] B. Widrow and M.E. Hoff, Adaptive switching circuits, *IRE WESCON Convention Record* **(4),** 1960, 96-104.

[Wilson and Martinez, 1997] D. R. Wilson and T. R. Martinez, Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research* **6**(1), 1997, 1-34.

[Witten and Frank, 2005] I. H. Witten and E. Frank, *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*, second edition, Morgan Kaufmann Publishers, USA, 2005.

[Yu and Ji, 2002] K. Yu and L. Ji, Karyotyping of Comparative Genomic Hybridization Human Metaphases Using Kernel Nearest-Neighbor Algorithm, *Cytometry*, **48**, 2002, 202–208.