**Name:** Aditya Somani **Roll No:** BE1851061 **PRN:** 71901204L

**To study about Supervised Learning – Classification**

|  |
| --- |
| **Aim** |
| **Implement Naïve Bayes Classifier and K-Nearest Neighbor Classifier on Data set of**  **your choice. Test and Compare for Accuracy and Precision.** |

|  |  |
| --- | --- |
| **Objective(s)** | |
| 1 | To implement Naïve Base classifier on a data set. |
| 2 | To implement K-nearest neighbor classifier on a data set. |
| 3 | To test accuracy & Precision. |

|  |
| --- |
| **Theory** |
| **Classification:**  In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.  Classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance. An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.  Classification and clustering are examples of the more general problem of pattern recognition, which is the assignment of some sort of output value to a given input value. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); parsing, which assigns a parse tree to an input sentence, describing the syntactic structureof the sentence; etc.  A common subclass of classification is probabilistic classification. Algorithms of this nature use statistical inference to find the best class for a given instance. Unlike other algorithms, which simply output a "best" class, probabilistic algorithms output a probability of the instance being a member of each of the possible classes. The best class is normally then selected as the one with the highest probability. However, such an algorithm has numerous advantages over non-probabilistic classifiers:   * It can output a confidence value associated with its choice (in general, a classifier that can do this is known as a *confidence-weighted classifier*). * Correspondingly, it can *abstain* when its confidence of choosing any particular output is too low. * Because of the probabilities which are generated, probabilistic classifiers can be more effectively incorporated into larger machine-learning tasks, in a way that partially or completely avoids the problem of *error propagation*.   **Naïve Bayse Classifier:**  In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying  Bayes' theorem with strong (naive) independence assumptions between the features.  Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrievalcommunity in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.  Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.  In the statistics and computer science literature, Naive Bayes models are known under a variety of names, including simple Bayesand independence Bayes.All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method;Russell and Norvig note that "[naive Bayes] is sometimes called a Bayesian classifier, a somewhat careless usage that has prompted some Bayesians to call it the idiot Bayes model.  For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.  **K-Nearest Neighbors algorithm**  In pattern recognition, the ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification)and [regression](https://en.wikipedia.org/wiki/Regression_analysis). In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression.  In k-NN *classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If*k* = 1, then the object is simply assigned to the class of that single nearest neighbor.  K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.  Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.  KNN is an *non parametric lazy learning* algorithm. That is a pretty concise statement. When you say a technique is non parametric , it means that it does not make any assumptions on the underlying data distribution. This is pretty useful , as in the real world , most of the practical data does not obey the typical theoretical assumptions made (eg gaussian mixtures, linearly separable etc) . Non parametric algorithms like KNN come to the rescue here.  It is also a lazy algorithm. What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase*or it is very minimal. This means the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. (Well this is an exaggeration, but not far from truth). This is in contrast to other techniques like SVM where you can discard all non support vectors without any problem.  Most of the lazy algorithms – especially KNN – makes decision based on the entire training data set (in the best case a subset of them).  The dichotomy is pretty obvious here – There is a non existent or minimal training phase but a costly testing phase. The cost is in terms of both time and memory. More time might be needed as in the worst case, all data points might take point in decision. More memory is needed as we need to store all training data. |
| Conclusion |
| Naïve Base & K-Nearest Neighbors algorithms have been successfully implemented & compared on same data set. |

import weka.core.Instances;

import weka.classifiers.Evaluation;

import weka.classifiers.bayes.NaiveBayes;

import java.io.BufferedReader;

import java.io.FileReader;

import java.lang.Exception;

public class NbClassifier {

public static void main(String[] args) {

// NbClassifier nb = new NbClassifier();

try {

BufferedReader br1 = null;

br1 = new BufferedReader(new FileReader("Datasets/IDSTr.arff"));

BufferedReader br2 = null;

br2 = new BufferedReader(new FileReader("Datasets/IDSTest.arff"));

Instances train = new Instances(br1);

Instances test = new Instances(br2);

train.setClassIndex(train.numAttributes() - 1);

test.setClassIndex(test.numAttributes() - 1);

// System.out.println("Training data: /n/n" +train);

// System.out.println("Test data: /n/n"+test);

NaiveBayes nB = new NaiveBayes();

nB.buildClassifier(train);

System.out.println("Naive Bayes Model :\n\n\n" + nB);

Evaluation eval = new Evaluation(test);

eval.evaluateModel(nB, test);

System.out.println("\nError Rate is :" + eval.errorRate());

System.out.println("\nAccuracy is :" + (1 - eval.errorRate()));

System.out.println("Precision" + " " + "Recall" + " " + "Fmeasure");

for (int i = 0; i < test.numClasses(); i++) {

System.out.println(eval.precision(i) + " " + eval.recall(i) + " " + eval.fMeasure(i));

}

} catch (Exception e) {

System.err.println(e.getMessage());

}

}

}