*.***CHAPTER 1**

**INTRODUCTION**

* 1. **OVERVIEW**

Data classification is one of the main tasks in pattern recognition. It aims to predict a label for every object accordingto the class that it belongs to, using a classification modelthat has been built from a training set. Usually, objects inclassification systems are represented by features, also calledattributes. In many predictive modelling applications for someobjects not contain all attribute values may be given. There areseveral reasons why this can happen, such as delays or failuresin communication lines, noise, measurement costs, or simply because they do not exist. In biomedical applications e.g., it isvery common that medical records have some lacking values,or in telemedicine data may arrive incomplete. In urgent cases,there might be no time to perform other tests.

Generally, the lack of attributes generates statistical variationsin the data and consequently deterioration in the classificationmodel and, as a consequence, classification accuracy is reduced. Furthermore, incomplete objects hamper data handling and analysis. Bias may result from differences betweenmissing and complete data. Most techniques, proposed inthe literature, to classify data with missing attributes employthe representation in the feature space.Three methods arecommonly used in this space to handle objects with missingattributes:

* Skipping incomplete objects

Skipping incomplete objects simply discards the incompleteobjects in the dataset in order to create a newcomplete dataset. It is application dependent whetherthis is an option.

* Imputation

This is probably the most frequently usedapproach. It estimates a value from the entire dataset to fill the missing attribute. Most common imputationtechniques are mean, median, random and Hot deck. In [statistics](https://en.wikipedia.org/wiki/Statistics), imputation is the process of replacing [missing data](https://en.wikipedia.org/wiki/Missing_data) with substituted values. When substituting for a data point, it is known as "unit imputation"; when substituting for a component of a data point, it is known as "item imputation". Because missing data can create problems for analyzing data, imputation is seen as a way to avoid pitfalls involved with [listwise deletion](https://en.wikipedia.org/wiki/Listwise_deletion) of cases that have missing values. That is to say, when one or more values are missing for a case, most [statistical packages](https://en.wikipedia.org/wiki/List_of_statistical_packages) default to discarding any case that has a missing value, which may introduce [bias](https://en.wikipedia.org/wiki/Bias_(statistics)) or affect the representativeness of the results. Imputation preserves all cases by replacing missing data with an estimated value based on other available information. Once all missing values have been imputed, the data set can then be analysed using standard techniques for complete data.

* Projection

Projection is the casein which the space is reduced to onedimension less for each missing attribute. This requiresa special computation of the classifier in the reducedspace. Random projection is a graphical technique used to [reduce the dimensionality](https://en.wikipedia.org/wiki/Dimensionality_reduction) of a set of points which lie in [Euclidean space](https://en.wikipedia.org/wiki/Euclidean_space). Random projection methods are powerful methods known for their simplicity and less erroneous output compared with other methods. According to experimental results, random projection preserve distances well, but empirical results are sparse. Projection basically is reducing number of random variables using various machine learning methods and techniques. Dimensionality reduction is used mainly to reduce the problem of managing and manipulation of large data sets. When we have large data sets it is too difficult for us to perform various operations like pattern recognition. Dimensionality reduction techniques generally uses linear transformations in determining the intrinsic dimensionality of the manifold as well as extracting its principal directions. There are various techniques like [Principal Component Analysis](https://en.wikipedia.org/wiki/Principal_Component_Analysis), [linear discriminant analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis), [canonical correlation analysis](https://en.wikipedia.org/wiki/Canonical_correlation_analysis), Discrete Cosine transform Method, Gauss Method, random Projectionetc.

Generally, methods for incomplete data classification basedon features require additional computations like imputation orclassifier updating. Consequently, these methods tend to be

computationally expensive.

* 1. **MOTIVATION**

Generally, methods for incomplete data classification basedon features require additional computations like imputation orclassifier updating. Consequently, these methods tend to be

computationally expensive. Hence, use of a couple of classifiers for additional computations would reduce the complexity and hence allow the results to be less expensive.

* 1. **OBJECTIVE**

To use a list of classifiers such as Naïve Bayes Classifier, K – Nearest Neighbour , Fisher classifier upon the subsequent data sets dividing them in a group of training and test sets and classifying the observations according to the training set values based on their algorithms.

* 1. **PLATFORM**

**Software Requirements**

* [R version 2.11.](http://cran.r-project.org/src/base/R-3)1 (or higher)
* R Studio Desktop 0.99.489
* Microsoft Excel 2007 (or higher)

**Hardware Requirements**

* Pentium 4 or above.
* RAM 1 GB or above.
* 10 GB Hard Disk or above

**CHAPTER 2**

**STATISTICAL CLASSIFICATION**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [statistics](https://en.wikipedia.org/wiki/Statistics), classification is the problem of identifying to which of a set of [categories](https://en.wikipedia.org/wiki/Categorical_data) (sub-populations) a new[observation](https://en.wikipedia.org/wiki/Observation) belongs, on the basis of a [training set](https://en.wikipedia.org/wiki/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"](https://en.wikipedia.org/wiki/Spam_filtering) 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.).

A classification is an ordered set of related categories used to group data according to its similarities. It consists of codes and descriptors and allows survey responses to be put into meaningful categories in order to produce useful data.

A classification is a useful tool for anyone developing statistical surveys. It is a framework which both simplifies the topic being studied and makes it easy to categorise all data or responses received.

In the terminology of machine learning,[[1]](https://en.wikipedia.org/wiki/Statistical_classification#cite_note-1) classification is considered an instance of [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning), i.e. learning where a training set of correctly identified observations is available. The corresponding [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning) procedure is known as [clustering](https://en.wikipedia.org/wiki/Cluster_analysis), and involves grouping data into categories based on some measure of inherent similarity or [distance](https://en.wikipedia.org/wiki/Distance).

Often, the individual observations are analyzed into a set of quantifiable properties, known variously as [explanatory variables](https://en.wikipedia.org/wiki/Explanatory_variables) or features. These properties may variously be [categorical](https://en.wikipedia.org/wiki/Categorical_data) (e.g. "A", "B", "AB" or "O", for [blood type](https://en.wikipedia.org/wiki/Blood_type)), [ordinal](https://en.wikipedia.org/wiki/Ordinal_data) (e.g. "large", "medium" or "small"), [integer-valued](https://en.wikipedia.org/wiki/Integer) (e.g. the number of occurrences of a part word in an [email](https://en.wikipedia.org/wiki/Email)) or [real-valued](https://en.wikipedia.org/wiki/Real_number) (e.g. a measurement of [blood pressure](https://en.wikipedia.org/wiki/Blood_pressure)). Other classifiers work by comparing observations to previous observations by means of a [similarity](https://en.wikipedia.org/wiki/Similarity_function) or [distance](https://en.wikipedia.org/wiki/Metric_(mathematics)) function.

An algorithm that implements classification, especially in a concrete implementation, is known as a [classifier](https://en.wikipedia.org/wiki/Pattern_recognition). The term "classifier" sometimes also refers to the mathematical [function](https://en.wikipedia.org/wiki/Function_(mathematics)), implemented by a classification algorithm, that maps input data to a category.

Terminology across fields is quite varied. In [statistics](https://en.wikipedia.org/wiki/Statistics), where classification is often done with [logistic regression](https://en.wikipedia.org/wiki/Logistic_regression) or a similar procedure, the properties of observations are termed [explanatory variables](https://en.wikipedia.org/wiki/Explanatory_variable) (or [independent variables](https://en.wikipedia.org/wiki/Independent_variable), regressors etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable). In machine learning, the observations are often known as instances, the explanatory variables are termed features (grouped into a[feature vector](https://en.wikipedia.org/wiki/Feature_vector)), and the possible categories to be predicted are classes. There is also some argument over whether classification methods that do not involve a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) can be considered "statistical". Other fields may use different terminology: e.g. in [community ecology](https://en.wikipedia.org/wiki/Community_ecology), the term "classification" normally refers to [cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis), i.e. a type of [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning), rather than the supervised learning described in this article.

**2.1 Features of a classification:**

It is generally accepted that a standard classification will usually meet a number of requirements which are outlined below.

* Exhaustive categories  - All survey responses need to fit into the classification structure somewhere
* Precise and meaningful descriptors for categories - The content of each category in the classification should be clearly defined
* Conceptually sound - The classification should have a conceptual basis and a logical structure
* Statistically balanced - In general, survey responses should not fall heavily into one category and sparsely into the other categories
* Operationally feasible - There is no point in having a classification that cannot be implemented in practice
* Statistically robust - The classification should be able to be used for a number of years without revision
* Internationally comparable - The classification should be comparable with any international standard classification

**2.2 Classification Algorithms**

A large number of [algorithms](https://en.wikipedia.org/wiki/Algorithm) for classification can be phrased in terms of a [linear function](https://en.wikipedia.org/wiki/Linear_function) that assigns a score to each possible category *k* by [combining](https://en.wikipedia.org/wiki/Linear_combination) the feature vector of an instance with a vector of weights, using a [dot product](https://en.wikipedia.org/wiki/Dot_product). The predicted category is the one with the highest score. This type of score function is known as a [linear predictor function](https://en.wikipedia.org/wiki/Linear_predictor_function) and has the following general form:

\operatorname{score}(\mathbf{X}_i,k) = \boldsymbol\beta_k \cdot \mathbf{X}_i,

where **X***i* is the feature vector for instance *i*, **β***k* is the vector of weights corresponding to category *k*, and score(**X***i*, *k*) is the score associated with assigning instance *i* to category*k*. In [discrete choice](https://en.wikipedia.org/wiki/Discrete_choice) theory, where instances represent people and categories represent choices, the score is considered the [utility](https://en.wikipedia.org/wiki/Utility) associated with person *i* choosing category *k*.

Algorithms with this basic setup are known as [linear classifiers](https://en.wikipedia.org/wiki/Linear_classifier). What distinguishes them is the procedure for determining (training) the optimal weights/coefficients and the way that the score is interpreted.

Examples of such algorithms are:

* [Logistic regression](https://en.wikipedia.org/wiki/Logistic_regression) and [Multinomial logistic regression](https://en.wikipedia.org/wiki/Multinomial_logistic_regression)
* [Probit regression](https://en.wikipedia.org/wiki/Probit_regression)
* The [perceptron](https://en.wikipedia.org/wiki/Perceptron) algorithm
* [Support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine)
* [Linear discriminant analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis).

Other examples of classification algorithms include :

Examples of classification algorithms include:

* [Linear classifiers](https://en.wikipedia.org/wiki/Linear_classifier)
  + [Fisher's linear discriminant](https://en.wikipedia.org/wiki/Fisher%27s_linear_discriminant)
  + [Logistic regression](https://en.wikipedia.org/wiki/Logistic_regression)
  + [Naive Bayes classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
  + [Perceptron](https://en.wikipedia.org/wiki/Perceptron)
* [Support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine)
  + [Least squares support vector machines](https://en.wikipedia.org/wiki/Least_squares_support_vector_machine)
* [Quadratic classifiers](https://en.wikipedia.org/wiki/Quadratic_classifier)
* [Kernel estimation](https://en.wikipedia.org/wiki/Variable_kernel_density_estimation#Use_for_statistical_classification)
  + [k-nearest neighbor](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)
* [Boosting (meta-algorithm)](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm))
* [Decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning)
  + [Random forests](https://en.wikipedia.org/wiki/Random_forest)
* [Neural networks](https://en.wikipedia.org/wiki/Artificial_neural_networks)
* [Learning vector quantization](https://en.wikipedia.org/wiki/Learning_vector_quantization)

**CHAPTER 3**

**NAIVE BAYES CLASSIFIER**

**3.1 Introduction**

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

Depending on the precise nature of the probability model, 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 believing in Bayesian probability or using any Bayesian methods.

In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers.Still, a comprehensive comparison with other classification methods in 2006 showed that Bayes classification is outperformed by more current approaches, such as boosted trees or random forests. An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

**3.2 THE NAIVE BAYES PROBABILISTIC MODEL**

Abstractly, the probability model for a classifier is a conditional model



over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables through . The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, we write



In plain English the above equation can be written as



In practice we are only interested in the numerator of that fraction, since the denominator does not depend on C and the values of the features are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model



which can be rewritten as follows, using repeated applications of the definition of conditional probability:









Now the "naive" conditional independence assumptions come into play: assume that each feature is conditionally independent of every other feature for j ≠ i. This means that



for I ≠ j and so the joint model can be expressed as

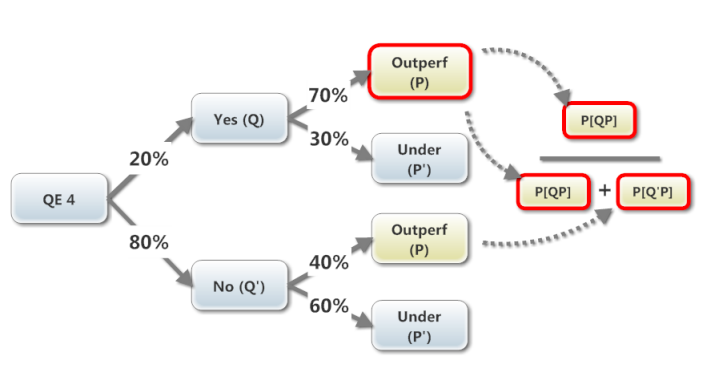




This means that under the above independence assumptions, the conditional distribution over the class variable C can be expressed like this:



where Z (the evidence) is a scaling factor dependent only on …….. i.e., a constant if the values of the feature variables are known.



**Figure 1: Bayes Theorem Aproach**

Models of this form are much more manageable, since they factor into a so-called *class prior p(C)* and independent probability distributions p(/C) . If there are K classes and if a model for each p(/C=c) can be expressed in terms of r parameters, then the corresponding naive Bayes model has (*k* − 1) + *n r k* parameters. In practice, often k=2 (binary classification) and r=1 (Bernoulli variables as features) are common, and so the total number of parameters of the naive Bayes model is 2n+1 , where n is the number of binary features used for classification and prediction.

**3.3 Parameter estimation**

All model parameters (*i.e.*, class priors and feature probability distributions) can be approximated with relative frequencies from the training set. These are maximum likelihood estimates of the probabilities. A class' prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an estimate for the class probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). To estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set. If one is dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

For example, suppose the training data contains a continuous attribute,x . We first segment the data by the class, and then compute the mean and variance of x in each class. Let be the mean of the values in associated with class *c*, and let be the variance of the values in associated with class *c*. Then, the probability of some value given a class, P(x=v/c) , can be computed by plugging into the equation for a Normal distribution parameterized by and . That is,



Another common technique for handling continuous values is to use binning to discretize the values. In general, the distribution method is a better choice if there is a small amount of training data, or if the precise distribution of the data is known. The discretization method tends to do better if there is a large amount of training data because it will learn to fit the distribution of the data. Since naive Bayes is typically used when a large amount of data is available (as more computationally expensive models can generally achieve better accuracy), the discretization method is generally preferred over the distribution method.

**3.4 Naïve Bayes Classifier in R**

It Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

**Functions Available**

## S3 method for class 'formula'

naiveBayes(formula, data, laplace = 0, ..., subset, na.action = na.pass)

## Default S3 method:

naiveBayes(x, y, laplace = 0, ...)

## S3 method for class 'naiveBayes'

predict(object, newdata, type = c("class", "raw"), threshold = 0.001, eps = 0, ...)

**Arguments**

X A numeric matrix, or a data frame of categorical and/or numeric

variables.

y Class vector.

formula A formula of the form class ~ x1 + x2 + .... Interactions are not allowed.

Data Either a data frame of predictors (categorical and/or numeric) or

contingency table.

laplace positive double controlling Laplace smoothing. The default (0) disables

Laplace smoothing.

... Currently not used.

subset For data given in a data frame, an index vector specifying the cases to

be used in the training sample.

na.action A function to specify the action to be taken if NAs are found. The

default action is not to count them for the computation of the probability

factors.

Object An object of class "naiveBayes".

Newdata A dataframe with new predictors (with possibly fewer columns than the

training data). Note that the column names of newdata are matched

against the training data ones.

type If "raw", the conditional a-posterior probabilities for each class are

returned, and the class with maximal probability else.

threshold Value replacing cells with probabilities within eps range.

eps double for specifying an epsilon-range to apply laplace smoothing (to

replace zero or close-zero probabilities by theshold.)

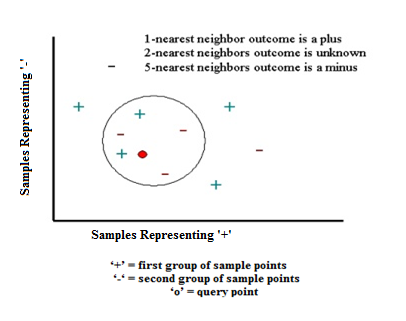
The standard naive Bayes classifier (at least this implementation) assumes independence of the predictor variables, and Gaussian distribution (given the target class) of metric predictors. For attributes with missing values, the corresponding table entries are omitted for prediction.

**CHAPTER 4**

**KNN CLASSIFICATION**

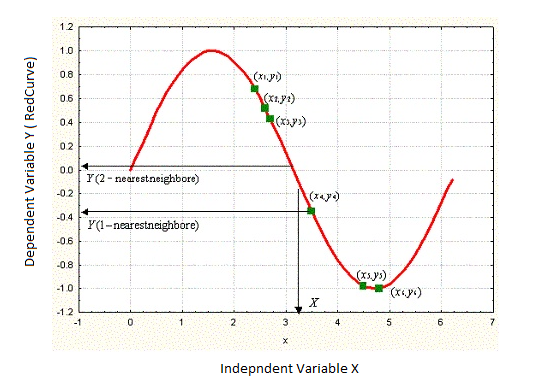
* 1. **Introduction**

To demonstrate a k-nearest neighbor analysis, let's consider the task of classifying a new object (query point) among a number of known examples. This is shown in the figure below, which depicts the examples (instances) with the plus and minus signs and the query point with a red circle. Our task is to estimate (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, we want to know whether the query point can be classified as a plus or a minus sign.



**Figure 2: Query Point Classification**

To proceed, let's consider the outcome of KNN based on 1-nearest neighbor. It is clear that in this case KNN will predict the outcome of the query point with a plus (since the closest point carries a plus sign). Now let's increase the number of nearest neighbors to 2, i.e., 2-nearest neighbors. This time KNN will not be able to classify the outcome of the query point since the second closest point is a minus, and so both the plus and the minus signs achieve the same score (i.e., win the same number of votes). For the next step, let's increase the number of nearest neighbors to 5 (5-nearest neighbors). This will define a nearest neighbor region, which is indicated by the circle shown in the figure above. Since there are 2 and 3 plus and minus signs, respectively, in this circle KNN will assign a minus sign to the outcome of the query point.

****

**Figure 3: Nearest Neighbour Estimation**

**4.1.1 Technical Details**

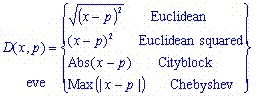
k-Nearest Neighbors (KNN) is a memory-based model defined by a set of objects known as examples (also known as instances) for which the outcome are known (i.e., the examples are labeled). Each example consists of a data case having a set of independent values labeled by a set of dependent outcomes. The independent and dependent variables can be either continuous or categorical. For continuous dependent variables, the task is regression; otherwise it is a classification. Thus, STATISTICA KNN can handle both regression and classification tasks.

Given a new case of dependent values (query point), we would like to estimate the outcome based on the KNN examples. STATISTICA KNN achieves this by finding K examples that are closest in distance to the query point, hence, the name k-Nearest Neighbors. For regression problems, KNN predictions are based on averaging the outcomes of the Knearest neighbors; for classification problems, a majority of voting is used.

The choice of K is essential in building the KNN model. In fact, k can be regarded as one of the most important factors of the model that can strongly influence the quality of predictions. One appropriate way to look at the number of nearest neighbours k is to think of it as a smoothing parameter. For any given problem, a small value of k will lead to a large variance in predictions. Alternatively, setting k to a large value may lead to a large model bias. Thus, k should be set to a value large enough to minimize the probability of misclassification and small enough (with respect to the number of cases in the example sample) so that the K nearest points are close enough to the query point. Thus, like any smoothing parameter, there is an optimal value for k that achieves the right trade off between the bias and the variance of the model. STATISTICA KNN can provide an estimate of K using an algorithm known as cross-validation.

## 4.1.2 Distance Metric

As mentioned before, given a query point, *KNN* makes predictions based on the outcome of the *K* neighbors closest to that point. Therefore, to make predictions with *KNN*, we need to define a metric for measuring the distance between the query point and cases from the examples sample. One of the most popular choices to measure this distance is known as Euclidean. Other measures include Euclidean squared, City-block, and Chebyshev.



where *x* and *p* are the query point and a case from the examples sample, respectively.

## 4.1.3 k-Nearest Neighbor Predictions

After selecting the value of *k*, you can make predictions based on the *KNN* examples. For regression, *KNN* predictions is the average of the *k-*nearest neighbours outcome.

http://www.statsoft.com/portals/0/Support/KNNOverViewEquationB.jpg

where *yi* is the *i*th case of the examples sample and *y* is the prediction (outcome) of the query point. In contrast to regression, in classification problems, *KNN* predictions are based on a voting scheme in which the winner is used to label the query.

 For binary classification tasks, odd values of *y = 1,3,5* are used to avoid ties, i.e., two classes labels achieving the same score.

So far we have discussed *KNN* analysis without paying any attention to the relative distance of the *K* nearest examples to the query point. In other words, we let the *K* eighbours have equal influence on predictions irrespective of their relative distance from the query point. An alternative approach (Shepard 1968) is to use arbitrarily large values of *K* (if not the entire prototype sample) with more importance given to cases closest to the query point. This is achieved using so-called distance weighting.

**4.2 K - NN Classifier in R**

### Description

k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

### Usage

knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)

### Arguments

|  |  |
| --- | --- |
| train | matrix or data frame of training set cases. |
| test | matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case. |
| cl | factor of true classifications of training set |
| k | number of neighbours considered. |
| l | minimum vote for definite decision, otherwise doubt. (More precisely, less than k-l dissenting votes are allowed, even if k is increased by ties.) |
| prob | If this is true, the proportion of the votes for the winning class are returned as attribute prob. |
| use.all | controls handling of ties. If true, all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours. |

### Value

Factor of classifications of test set. doubt will be returned as NA.

**EXAMPLE**

train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])

test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])

cl<- factor(c(rep("s",25), rep("c",25), rep("v",25)))

knn(train, test, cl, k = 3, prob=TRUE)

attributes(.Last.value)

**Description**

The K Nearest Neighbor Rule (k-NNR) is a very intuitive method for classifying unlabeled examples based on the similarity of these examples with other examples present in the training set.

If there is an unlabeled example xu∈ℜD, we have to find the k closest labeled examples in the training data set and assign xu to the class that appears most

frequently within the k-subset.

For k-NN classifier we require

1. An integer k
2. A set of labeled examples ie. the training data
3. A metric for measuring the closeness

Example :

Considering an example of three classes the goal is to find a class label for the unknown example xu.

For this purpose we use the Euclidean distance and a value of k=5 neighbours of the 5 closest neighbours, 4 belong to ω1 and 1 belongs to ω3, so xu is assigned to ω1, the predominant class.

KNN is a lazy learning algorithm since it defers the data processing until it receives a request for classifying an unlabelled example and it replies to a request for information by combining the stored training data and it discards any intermediate results and constructed answers.

KNN uses a stratergyoppsed to an eager learning algorithm that compliestha data into a compressed model or description and discards the training data after compilation of the model. The incoming patterns are classified using the induced model , which is retained for future requests.

KNN(Lazy algorithm) have fewer computations costs during training than eager algorithms.

But the lazy algorithms have higher computational costs at recall and requires greater storage.

**4.3 KNN Characteristics**

There are various advantages of KNN Algorithm

1. KNN is analytically tractable
2. It provides simple implementation
3. It is nearly optimal in the large sample limit(N🡪 infinity)
4. KNN uses local information which yields high adaptive behaviour
5. KNN Lends itself very easily to parallel implementation

Disadvantages of KNN Algorithm

1. KNN requires large storage system
2. Computation recall costs are intensively high
3. Highly susceptible to the curse of dimensionality

**CHAPTER-5**

**FISHER’S LINEAR DISCRIMINANT**

**5.1 Introduction**

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in [statistics](https://en.wikipedia.org/wiki/Statistics), [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition) and [machine learning](https://en.wikipedia.org/wiki/Machine_learning) to find a [linear combination](https://en.wikipedia.org/wiki/Linear_combination) of [features](https://en.wikipedia.org/wiki/Features_(pattern_recognition)) that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier), or, more commonly, for [dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction) before later [classification](https://en.wikipedia.org/wiki/Statistical_classification).

LDA is closely related to [analysis of variance](https://en.wikipedia.org/wiki/Analysis_of_variance) (ANOVA) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), which also attempt to express one [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable) as a linear combination of other features or measurements. However, ANOVA uses [categorical](https://en.wikipedia.org/wiki/Categorical_variable) [independent variables](https://en.wikipedia.org/wiki/Independent_variables) and a [continuous](https://en.wikipedia.org/wiki/Continuous_variable)[dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), whereas discriminant analysis has continuous [independent variables](https://en.wikipedia.org/wiki/Independent_variables) and a categorical dependent variable (*i.e.* the class label). [Logistic regression](https://en.wikipedia.org/wiki/Logistic_regression) and [probit regression](https://en.wikipedia.org/wiki/Probit_regression) are more similar to LDA than ANOVA is, as they also explain a categorical variable by the values of continuous independent variables. These other methods are preferable in applications where it is not reasonable to assume that the independent variables are normally distributed, which is a fundamental assumption of the LDA method.

LDA is also closely related to [principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis) (PCA) and [factor analysis](https://en.wikipedia.org/wiki/Factor_analysis) in that they both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Discriminant analysis is also different from factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made.

LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is [discriminant correspondence analysis](https://en.wikipedia.org/w/index.php?title=Discriminant_correspondence_analysis&action=edit&redlink=1).

**5.2 Different Approaches to LDA**

Data sets can be transformed and test vectors can be classified in the transformed space by two different approaches.

Class-dependent transformation: This type of approach involves maximizing the ratio of between

class variance to within class variance. The main objective is to maximize this ratio so that adequate class separability is obtained. The class-specific type approach involves using two optimizing criteria for transforming the data sets independently.

Class-independent transformation: This approach involves maximizing the ratio of overall variance to within class variance. This approach uses only one optimizing criterion to transform the data sets and hence all data points irrespective of their class identity are transformed using this.

**5.3 Mathematical Operations**

In this section, the mathematical operations involved in using LDA will be analyzed the aid of sample set .

# 1.To find the discriminant: D=a1\*X1+a2\*X2+………ai\*Xi Where D=discriminant function Xi=No. of discriminant variables a=Discriminant Coefficients

2.Formulate the data sets and the test sets, which are to be classified in the original space. The given data sets and the test vectors are formulated, a graphical plot of the data sets and test vectors for the example considered in original space .

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  | |  |
|  | *a*11 | *a*12 | |  | *b*11 | *b*12 | |  | |  |
| *set*1 = | *a*21 | *a*22 | | *set*2 = | *b*21 | *b*22 | |  |
|  | |  |  | |  |  |
|  |  |  |  |  | |  |
|  |  | |  |  |  | |  |  | |  |
|  | *am*1*am*2 | | |  | *bm*1*bm*2 | | |  | |  |
|  |  |  |  |  |  |  |  |  | |  |

3.Compute the mean of each data set and mean of entire data set. Let 1and 2 be the mean of set 1 and set 2 respectively and 3 be mean of entire data, which is obtained by merging set 1 and set 2.

3 =*p*11 + *p*22

where p1 and p2 are the probabilities of the classes. In the case of this simple two

class problem, the probability factor is assumed to be 0.5.

4.In LDA, within-class and between-class scatter are used to formulate criteria for class

separability. Within-class scatter is the expected covariance of each of the classes.

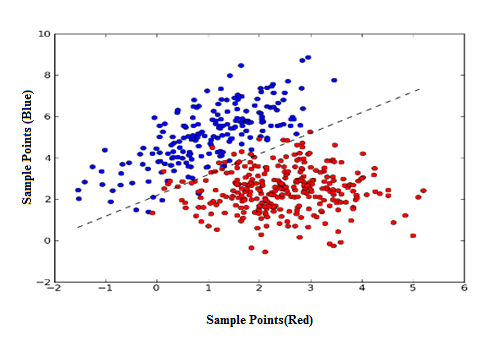
*Sw=**p j* *covj*

Therefore, for the two-class problem,

*Sw*= 0.5 *cov*1 + 0.5 *cov*2

All the covariance matrices are symmetric. Let cov1 and cov2 be the covariance of set 1 and set 2 respectively. Covariance matrix is computed using the following equation.

*covj=* ***x*** *j* – *j****x*** *j* – *j**T*

**

**Figure 4:LDA Predictions Among Given Samples**

**5.4 LDA Classifier in R**

**Description**

The **MASS** package contains functions for performing linearand quadratic discriminant function analysis. Unless prior probabilities are specified, each assumes proportional prior probabilities (i.e., prior probabilities are based on sample sizes). In the examples below, **lower case**letters are numeric variables and **upper case letters** are categorical [factors](http://www.statmethods.net/input/datatypes.html).

**Functions Available**

lda(x, ...)

## S3 method for class 'formula'

lda(formula, data, ..., subset, na.action)

## Default S3 method:

lda(x, grouping, prior = proportions, tol = 1.0e-4,

method, CV = FALSE, nu, ...)

## S3 method for class 'data.frame'

lda(x, ...)

## S3 method for class 'matrix'

lda(x, grouping, ..., subset, na.action)

**Arguments**

|  |  |
| --- | --- |
| formula | A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators. |
|  |  |
| data | Data frame from which variables specified in formula are preferentially to be taken. |
| prior | the prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels. |
| tol | A tolerance to decide if a matrix is singular; it will reject variables and linear combinations of unit-variance variables whose variance is less than tol^2. |
| subset | An index vector specifying the cases to be used in the training sample. |
| na.action | A function to specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.) |
| CV | If true, returns results (classes and posterior probabilities) for leave-one-out cross-validation. Note that if the prior is estimated, the proportions in the whole dataset are used. |

**Details**

The function tries hard to detect if the within-class covariance matrix is singular. If any variable has within-group variance less than tol^2 it will stop and report the variable as constant. This could result from poor scaling of the problem, but is more likely to result from constant variables.

Specifying the prior will affect the classification unless over-ridden in predict.lda. Unlike in most statistical packages, it will also affect the rotation of the linear discriminants within their space, as a weighted between-groups covariance matrix is used. Thus the first few linear discriminants emphasize the differences between groups with the weights given by the prior, which may differ from their prevalence in the dataset.

If one or more groups is missing in the supplied data, they are dropped with a warning, but the classifications produced are with respect to the original set of levels.

**5.5 Applications of LDA**

LDA is applied in [positioning](https://en.wikipedia.org/wiki/Positioning_(marketing)) and [product management](https://en.wikipedia.org/wiki/Product_management).

**Bankruptcy prediction**

In [bankruptcy prediction](https://en.wikipedia.org/wiki/Bankruptcy_prediction) based on accounting ratios and other financial variables, linear discriminant analysis was the first statistical method applied to systematically explain which firms entered bankruptcy vs. survived. Despite limitations including known non conformance of accounting ratios to the normal distribution assumptions of LDA.

**Face recognition**

In computerized [face recognition](https://en.wikipedia.org/wiki/Facial_recognition_system), each face is represented by a large number of pixel values. Linear discriminant analysis is primarily used here to reduce the number of features to a more manageable number before classification. Each of the new dimensions is a linear combination of pixel values, which form a template. The linear combinations obtained using Fisher's linear discriminant are called Fisher faces, while those obtained using the related [principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis) are called [eigen faces](https://en.wikipedia.org/wiki/Eigenfaces).

**Biomedical studies**

The main application of discriminant analysis in medicine is the assessment of severity state of a patient and prognosis of disease outcome. For example, during retrospective analysis, patients are divided into groups according to severity of disease – mild, moderate and severe form. Then results of clinical and laboratory analyses are studied in order to reveal variables which are statistically different in studied groups. Using these variables, discriminant functions are built which help to objectively classify disease in a future patient into mild, moderate or severe form.

**Earth Science**

This method can be used to separate the alteration zones. For example, when different data from various zones are available, discriminate analysis can find the pattern within the data and classify it effectively.

**Conclusion and Future Scope**

For the given dataset the proposed approaches show, in general, better results than those of the Projection technique and the Imputation method using SVR.

There are various advantages of using classifier over the previous techniques:

* Classifiers hardly need to be updated, since the original trained classifiers in the dissimilarity space can be used to classify any incomplete object regardless of which attribute is missing.
* Imputation techniques are not needed.
* As additional computations are not required, processing time is less.

From the three classifiers studied in the experiments, Fisher classifier shows better performance for the proposed techniques, whereas 1NN and NBC classifiers showed better results in the feature space. In spite of this, the proposed approaches using the Fisher classifier outperform other classifiers in the feature space. Moreover, the results obtained with both approaches for incomplete objects are better than the results for complete objects in the feature space.

Our results are based on just a single missing attribute. In case more attributes are missing, the obtained dissimilarities will be more severely affected. It has to be studied to what extend a correction can still be used and when a recomputation of the classifier has to be preferred.

**References**

[1] M. Millan-Giraldo,Robert P.W.Duin and J.S. Sanchez ,“Dissimilarity-based Classification of Data with Missing Attributes”,in *2nd International Workshop on Cognitive Information Processing* , vol. 3,pp. 293-298, 2010.

[2] S.Zhang,”Parimputation: from imputation and null-imputation to partially imputation”, in IEEE *Intelligent Informatics Bulletin*, vol. 9,pp. 32-38, 2008.

[3] Bruzzone, L. Poli, R. Serpico,”S.B.: An extension of Jeffereys Matusita distance to multiclass cases for feature selection”, in *IEEE* *Transaction on Geoscience and Remote Sensing*,vol. 33(6),pp. 13-18, 1995.

[4] E. Pekalska, Y.Y. Tang, S.P. Wang, G. Lorette, D.S. Yeung, H.Yan ,” Dissimilarity-based classification for vecto-rial representations”, in *IEEE Computer Society Press, Los* *Alamitos*, vol. 3, pp. 137-140, 2006.

[5] T. Delavallade and T.H. Dang, “Using Entropy to Impute Missing Data in a Classification Task”, in  *IEEE International Conference on Fuzzy Systems*, London, vol. 3, pp.1-6,2007.

[6] Y. Zhang, Y. Liu,”Data Imputation Using Least Squares Support Vector Machines in Urban Arterial Streets”,in *IEEE Signal Processing Letters*, vol.16 (5), pp. 414-417,2009.

[7] M. Saar-Tsechansky and F. Provost,”Handling missing values when applying classification Models”, in *Journal of Machine Learning Research*,vol.8,pp.1625–1657, 2007.

[8] F. Honghai, C. Guoshun, Y. Cheng, Y. Bingru and C. Yumei,”A SVM Regression

Based Approach to Filling in Missing Values”,in *LNCS* *- Knowledge-Based Intelligent*

*Information and Engineering Systems*, *Springer Berlin - Heidelberg*, vol. 3683,pp. 581-

587,2005.

[9] M. Friedman,”The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance”, *Journal of the American Statistical Association*, vol.32, pp.675-701,2008.

[10] Wong, K. C. and Chiu, K. Y.,” Synthesizing statistical knowledge for incomplete mixed-mode data”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*,vol.9,pp.796–805,2007.

[11] Slowinski R. and Vanderpooten D.,”A generalized definition of rough approximations based on similarity”,in *IEEE Transactions on Knowledge and Data Engineering*, vol.12 pp. 331–336,2009.

[12] Grzymala-Busse J.W.,”Rough set strategies to data with missing attribute values”,in *IEEE International Conference on Data Mining*, pp. 56–63, November 2003.

**Appendix**

**1.Naive bayes Classifier**

**#Naive Bayes Classifier**

library(e1071)

data(iris)

View(iris)

str(iris)

table(iris$Species)

head(iris)

set.seed(9850)

runif(5)

gp <- runif(nrow(iris))

gp

iris5 <- iris[order(gp),]

str(iris)

head(iris5)

head(iris5,10)

str(iris5)

summary(iris5[,c(1,2,3,4)])

normalize <- function(x) {

+return (( x- min(x)) / (max(x) - min(x) ))

}

normalize(c(1,2,3,4,5))

iris\_n <- as.data.frame(lapply(iris5[,c(1,2,3,4)], normalize))

str(iris\_n)

summary(iris\_n)

iris\_train <- iris\_n[1:129,]

iris\_test <- iris\_n[130:150,]

iris\_train\_target <- iris5[1:129, 5]

iris\_test\_target <- iris5[130:150, 5]

model <- naiveBayes(Species ~., data=iris,na.action.na.omit)

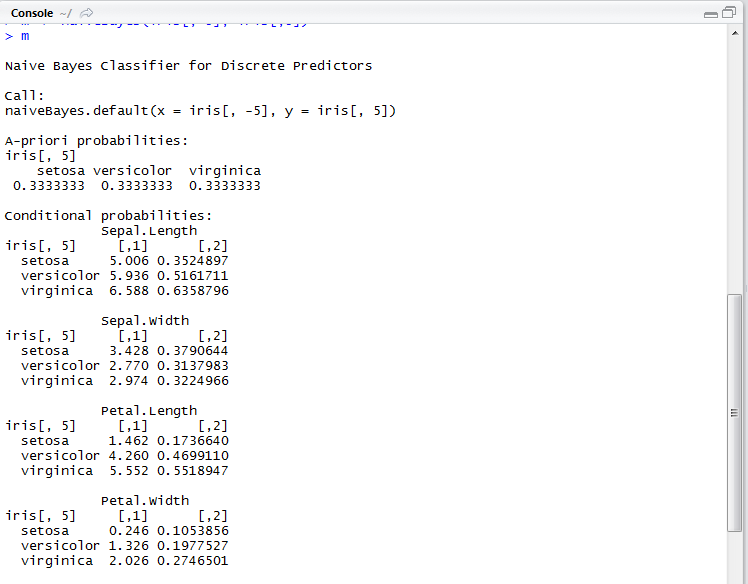
model <- naiveBayes(Species ~., data=iris,na.action.na.pass)

pred <- predict(model,iris)

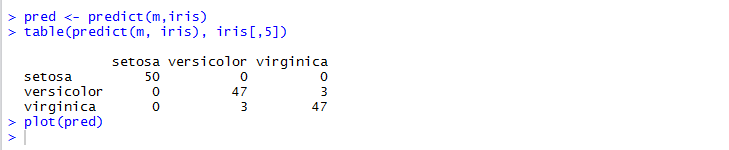
table(pred,iris$Species)

plot(pred,main=”Classification of Species”,xlab=”species”,ylab=”Number of Atrributes”)

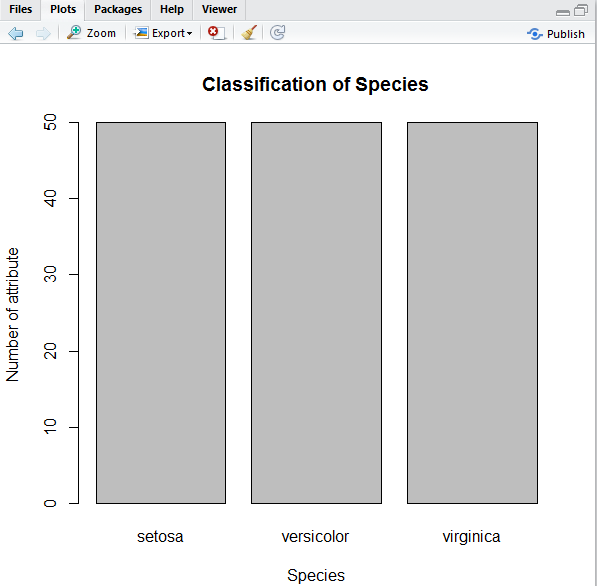
**Conditional Probabilities Output:**



**Output Matrix:**



**Output Plot :**



**2. Linear Discriminant Analysis**

library(MASS)

View(iris)

head(iris)

plot( iris[ , c(2,3)],

col = iris[,1])

iris.lda <- lda(Species ~ Petal.Length + Sepal.Width,

data = iris)

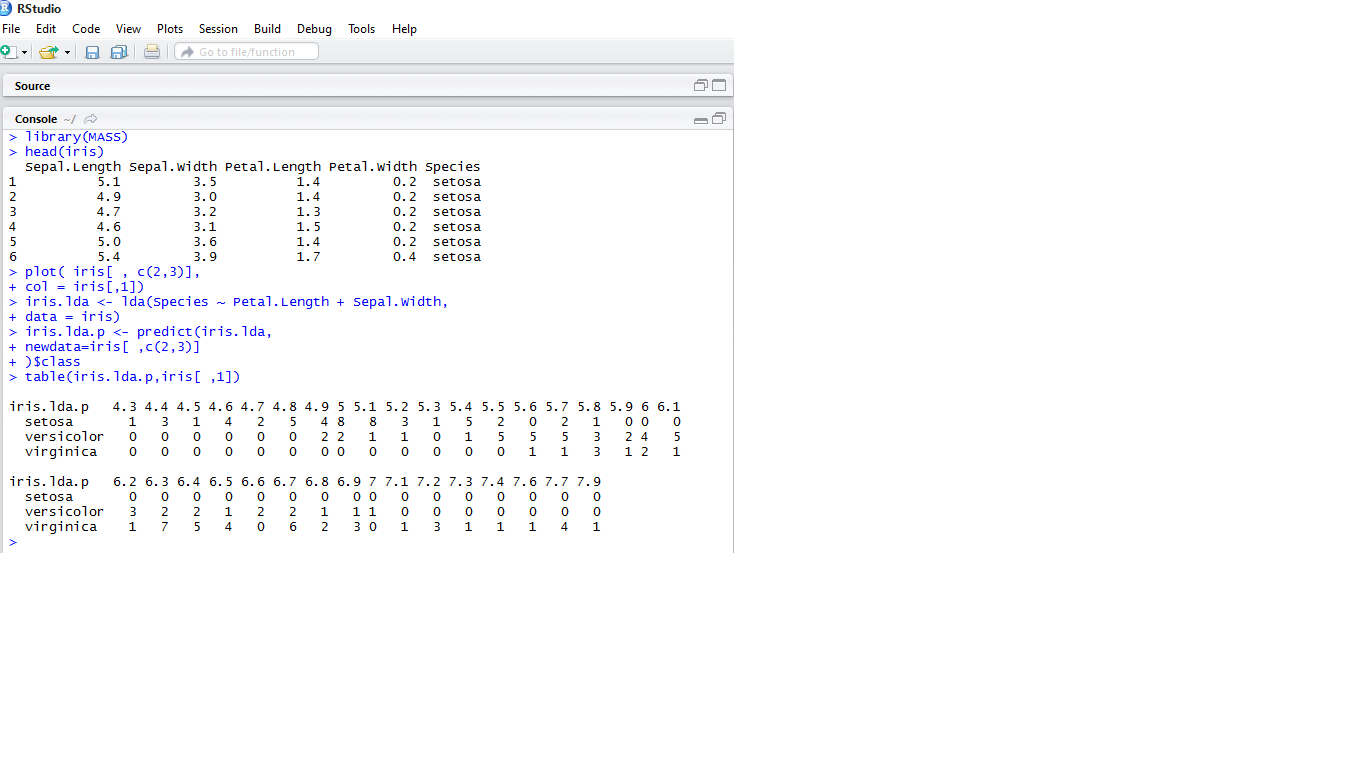
iris.lda.p <- predict(iris.lda,

newdata=iris[ ,c(2,3)]

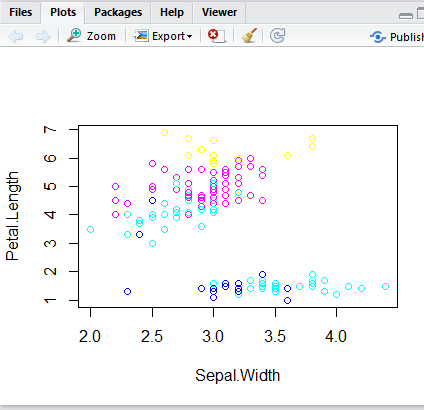
)$class

table(iris.lda.p,iris[ ,1])

**Output:**



**Output Plot:**



**3. K – Nearest Neighbour**

data(iris)

View(iris)

str(iris)

table(iris$Species)

head(iris)

set.seed(9850)

gp <- runif(nrow(iris))

gp

iris5 <- iris[order(gp),]

head(iris5)

str(iris5)

summary(iris5[,c(1,2,3,4)])

normalize <- function(x) {

+return (( x- min(x)) / (max(x) - min(x) ))

}

iris\_n <- as.data.frame(lapply(iris5[,c(1,2,3,4)], normalize))

str(iris\_n)

summary(iris\_n)

iris\_train <- iris\_n[1:129,]

iris\_test <- iris\_n[130:150,]

iris\_train\_target <- iris5[1:129, 5]

iris\_test\_target <- iris5[130:150, 5]

require(class)

m1 <- knn(train = iris\_train, test = iris\_test, cl = iris\_train\_target, k = 13 )

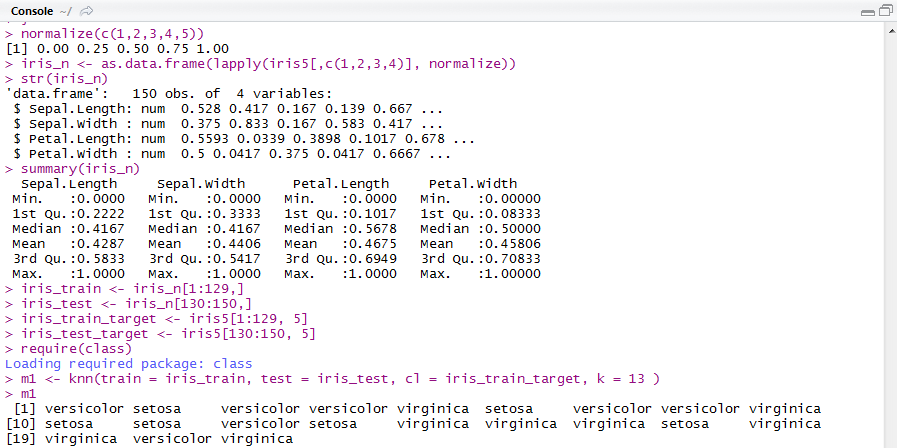
m1

table(iris\_test\_target, m1)

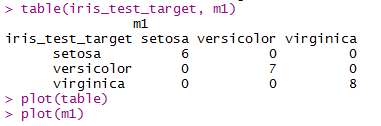
plot(table)

plot(m1,xlab=”Species”,ylab=”No. of Predictions”)

**Output Predictions:**

****

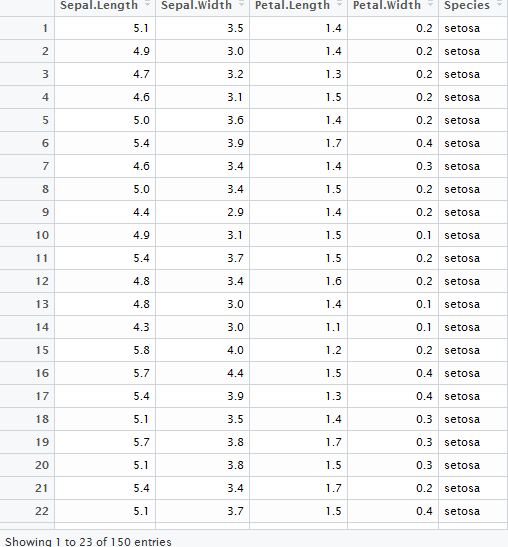
**Output Matrix :**

****

**Output Plot :**

****

1. **Data Set (iris)**

****