Machine Learning

tanshuqiu

Email: tsq@cqut.edu.cn

Pattern Recognition through Classification Algorithms

1. Predicting a response by decision trees
2. Probabilistic classification algorithms - Naive Bayes
3. Find similarities using nearest neighbor classifiers

1. Predicting a response by decision trees

Classification algorithms study how to automatically learn to make accurate predictions based on observations. Starting from a set of predefined class labels, the algorithm gives each piece of data input a class label in accordance with the training model. If there are just two distinction classes, we talk about binary classification; otherwise, we go for multi-class classification. In more detail, each category corresponds to a different label; the algorithm attaches a label to each instance, which simply indicates which class the data belongs to. A procedure that can perform this function is commonly called a classifier.

1. Predicting a response by decision trees

Classification has some analogy with regression, finding relationships between variables - regression techniques. As well as regression, classification uses known labels of a training dataset to predict the response of the new test dataset. The main difference between regression and classification is that regression is used to predict continuous values, whereas classification works with categorical data.

1. Predicting a response by decision trees

Predicting a response by decision trees

A decision tree is the graphic demonstration of a choice made or proposed. What seems most interesting is not always useful, and not always are things so clear that you can choose between two solutions immediately. Often, a decision is determined by a series of waterfall conditions. Expressing this concept with tables and numbers is difficult, and even if a table formally represents the phenomenon, it can confuse the reader because the justification of the choice is not immediately apparent.

1. Predicting a response by decision trees

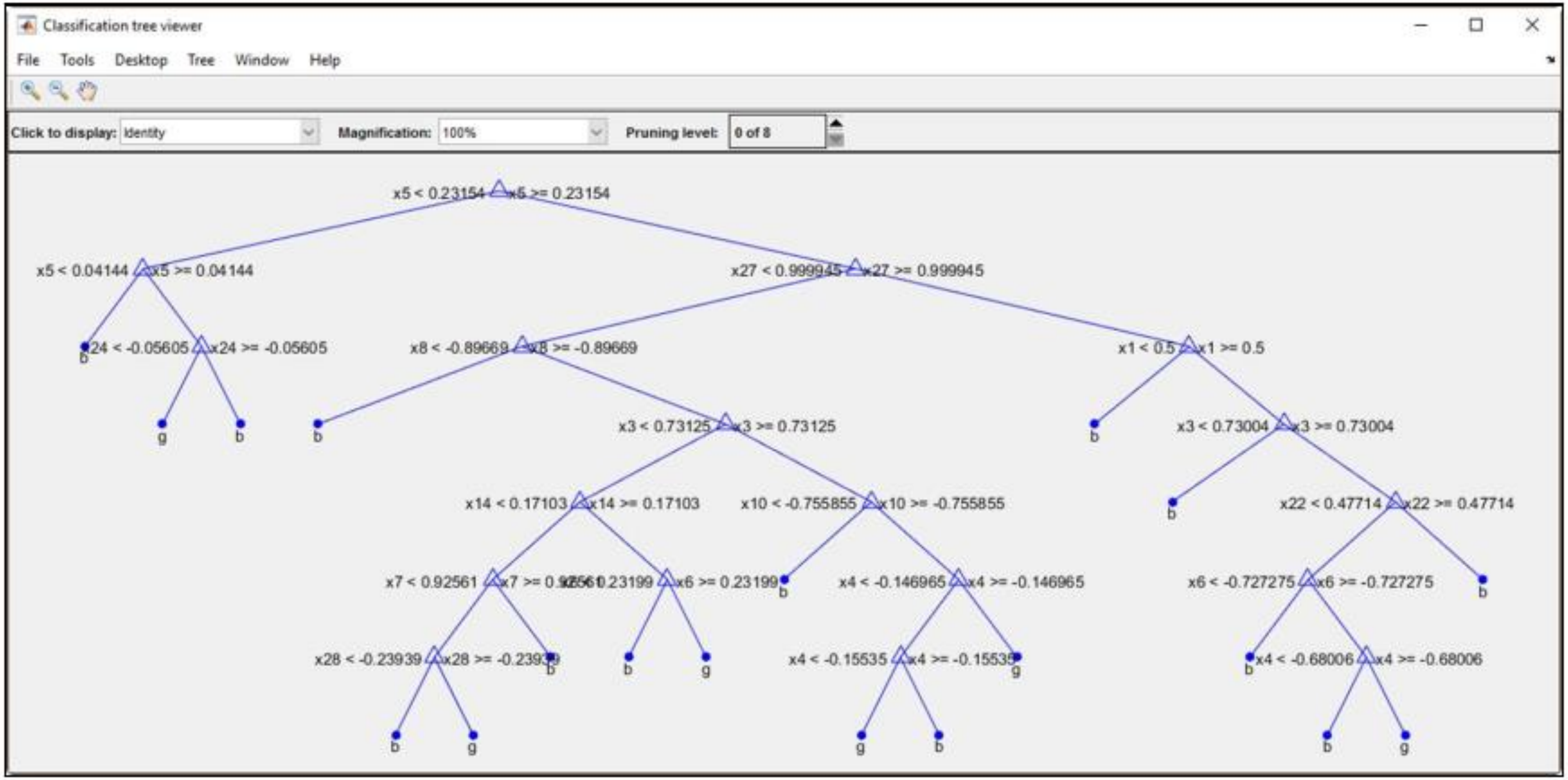
Predicting a response by decision trees

A decision tree consists of:

Nodes containing the names of independent variables

Branches labeled with the possible values of independent variables

Leaf nodes representing the classes, that is, collections of observations grouped according to the values of one independent variable and joined to nodes via branches



Click to display: Identity

Magnification: 100%

Pruning level: 0 of 8

1. Predicting a response by decision trees

There are cases where classification rules are **univariate**, in the sense that they consider a single predictor (target attribute) at a time. However, there are **multivariate** algorithms too, in which the predictor is represented by a linear combination of variables.

The subdivision produces a hierarchy tree, where the subsets are called nodes, and the final or terminal are called leaf nodes. Specifically, **nodes** are labeled with the attribute name, **branches** are labeled with the possible values of the above attribute, and **leaf nodes** are labeled with the different values of the target attribute. I mean, the values that describe the membership classes.

1. Predicting a response by decision trees

An object is classified by following a path along the tree that leads from the root to a leaf. The paths represent the rules of classification or production rules. The branches are the values assumed by the different attributes. The leaves are theclassifications. The rule is written along the tree from the node to the different leaves. All possible paths represent the possible rules.

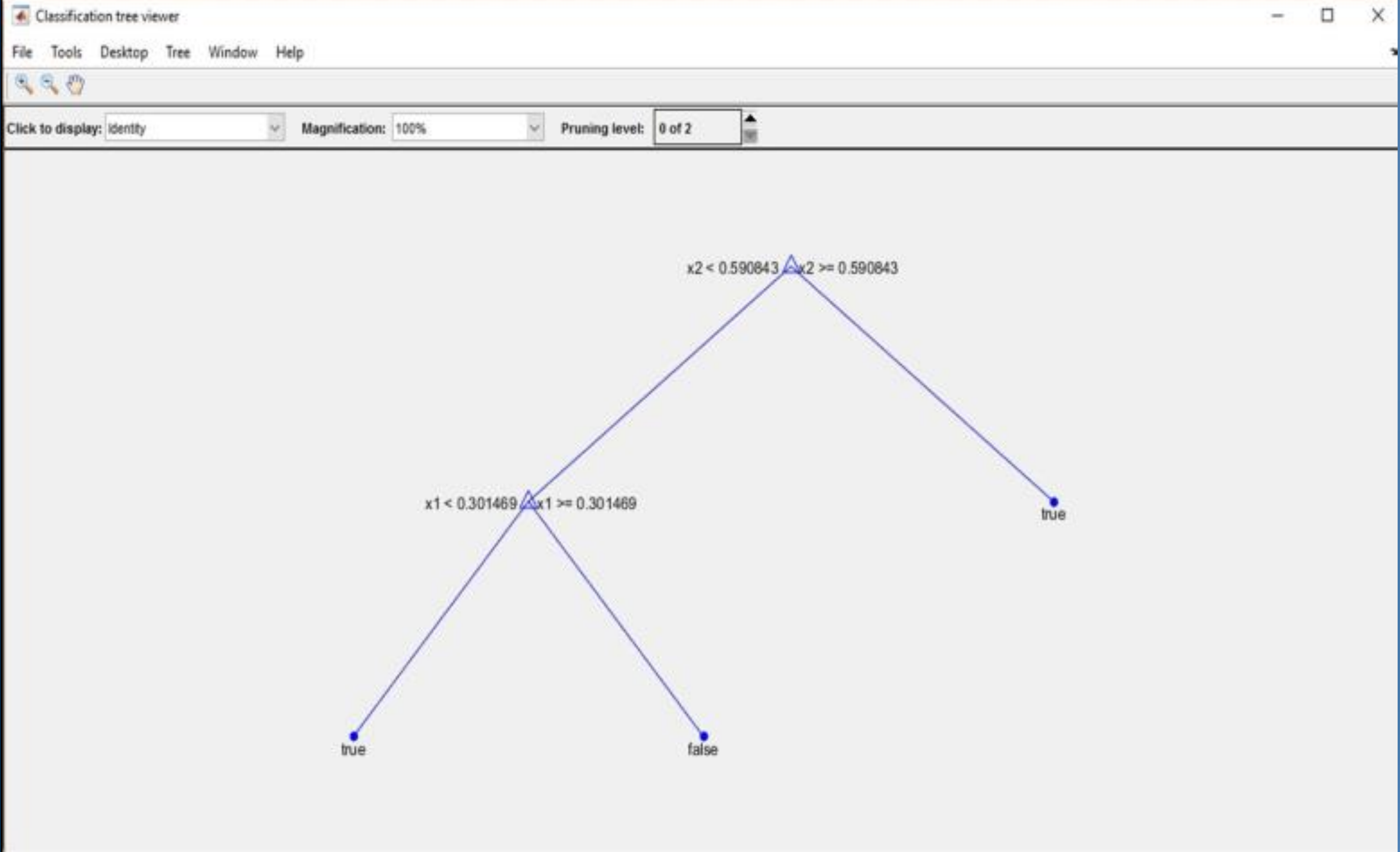
1. Predicting a response by decision trees

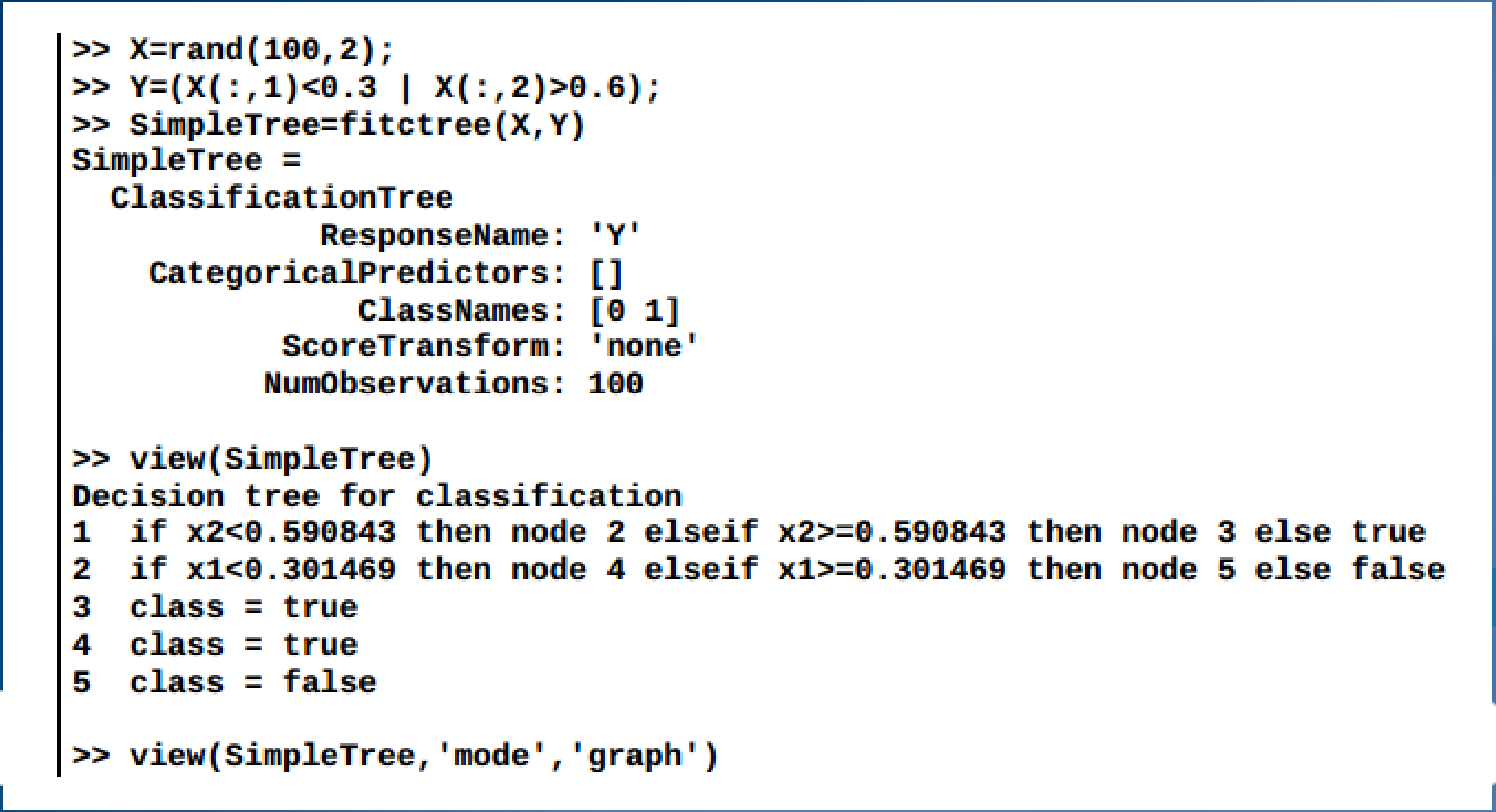
To classify an instance with a decision tree, you have to follow the given steps:

1. Start from the root.
2. Select the instance attribute associated with the current node.
3. Follow the branch associated with the value assigned to that attribute in the instance.
4. If you reach a leaf, return the label associated with the leaf; otherwise, beginning with the current node, repeat from step 2.

1. Predicting a response by decision trees

To understand these processes, analyze a simple classification tree. This tree predicts classifications based on two predictors, x1 and x2. We will classify a predictor as true when $x1 < 0.3$, $x2 > 0.6$, and false otherwise.


Classification tree viewer
File Tools Desktop Tree Window Help
Click to display: Identity
Magnification: 100%
Pruning level: 0 of 2
x2 < 0.590843
x2 >= 0.590843
x1 < 0.301469
x1 >= 0.301469
true
false
true



```
>> X=rand(100,2);
>> Y=(X(:,1)<0.3 | X(:,2)>0.6);
>> SimpleTree=fitctree(X,Y)
SimpleTree =
  ClassificationTree
           ResponseName: 'Y'
  CategoricalPredictors: []
             ClassNames: [0 1]
         ScoreTransform: 'none'
        NumObservations: 100

>> view(SimpleTree)
Decision tree for classification
1  if x2<0.590843 then node 2 elseif x2>=0.590843 then node 3 else true
2  if x1<0.301469 then node 4 elseif x1>=0.301469 then node 5 else false
3  class = true
4  class = true
5  class = false

>> view(SimpleTree,'mode','graph')
```

2. Probabilistic classification algorithms - Naive Bayes

Bayesian classification is a statistical technique that determines the probability that an element belongs to a particular class. For example, this technique can be used to estimate the probability of a customer belonging to the class of sports car buyers, given some customer attributes such as type of work performed, age, income, civil status, sports practiced, and so on.

2. Probabilistic classification algorithms - Naive Bayes

The technique is based on the theorem of Bayes, a mathematician and British Presbyterian minister of the eighteenth century. The theorem defines the posterior probability of an event with respect to another. Posterior, in this context, means after taking into account the events relevant to the particular case being examined as if they have already happened.

2. Probabilistic classification algorithms - Naive Bayes

The Bayesian classifier algorithm assumes that the effect of an event on a given class is independent of the values of other events. This assumption, called the conditional independence of the classes, is intended to simplify calculations, and for this reason the algorithm is named naive. When this assumption is true in reality, the accuracy of the algorithm is comparable to others.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

In a bag, there are 7 white balls and 3 black balls. Except in color, the balls are identical; they are made of the same material, they are the same size, they are perfectly spherical, and so on. I'll put a hand in the bag without looking inside, pulling out a random ball. What is the probability that the pulled out ball is black?

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

The probability (a priori) that a given event (E) occurs is the ratio between the number (s) of favorable cases of the event itself and the total number (n) of the possible cases, provided all considered cases are equally probable:

$$P = P(E) = \frac{number\ of\ favorable\ cases}{total\ number\ of\ possible\ cases} = \frac{s}{n}$$

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

Let's look at two simple examples:

By throwing a coin, what is the probability that it shows a head?

The possible cases are 2, heads and tails {H, T}, the favorable cases are 1{H}.

By throwing a dice, what is the probability that 5 is out?

The possible cases are 6, {1, 2, 3, 4, 5, 6}, and the favorable cases are 1{5}.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

To define probability, use the concept of equally likely events. It is therefore necessary to clarify what is meant by equally likely events.

Given a group of events, if there are no valid reasons to think that some event occurs more or less easily than others, then all group events must be considered equally likely.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

The probability of an event P(E) is always a number between 0 and 1:

$$0 \leq P(E) \leq 1$$

An event that has probability 0 is called an impossible event.
An event that has probability 1 is called a certain event.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

An event that has probability 0 is called an impossible event. Suppose we have six red balls in a bag, what is the probability of picking a black ball? The possible cases are 6; the favorable cases are 0 because there are no black balls in the bag. P(E) = 0/6 = 0.

An event that has probability 1 is called a certain event. Suppose we have six red balls in a bag, what is the probability of picking a red ball? The possible cases are 6; the favorable cases are 6 because there are only red balls in the bag. P(E) = 6/6 = 1.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

The classical definition of probability, based on a discrete and finite number of events, is hardly extendable to the case of continuous variables. The ideal condition of perfect uniformity, where all possible outcomes (the space of events) are previously known and all are equally probable, is a weak element of that definition.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

The classical definition of probability, based on a discrete and finite number of events, is hardly extendable to the case of continuous variables. The ideal condition of perfect uniformity, where all possible outcomes (the space of events) are previously known and all are equally probable, is a weak element of that definition.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

An important advance compared to the classic concept in which probability is established a priori, before looking at the data, is contained in the frequentist definition of the probability; this is instead obtained later, after examining the data.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

So far, we've talked about the likelihood of an event, but what happens when the possible events are more than one?

Two random events, A and B, are independent if the probability of the occurrence of event A is not dependent onwhether event B has occurred, and vice versa. For example, if we have two 52 decks of French playing cards. When extracting a card from each deck, the following two events are independent:

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

E1 = The card extracted from the first deck is an ace.

E2 = The card extracted from the second deck is a clubs card.

The two events are independent; each can happen with the same probability independently of the other's occurrence.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

Conversely, a random event, A, is dependent on another event, B, if the probability of event A depends on whether event B has occurred or not. Suppose we have a deck of 52 cards. By extracting two cards in succession without put the first card back in the deck, the following two events are dependent:

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

E1 = The first extracted card is an ace.

E2 = The second extracted card is an ace.

To be precise, the probability of E2 depends on whether or not E1 occurs.

Indeed:

The probability of E1 is 4/52

The probability of E2 if the first card was an ace is 3/51

The probability of E2 if the first card was not an ace is 4/51

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

Let us now deal with other cases of mutual interaction between events. Accidental events that cannot occur simultaneously on a given trial are considered mutually exclusive or disjoint. By extracting a card from a deck of 52, the following two events are mutually exclusive:

E1 = The ace of hearts comes out

E2 = One face card comes out

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

Indeed, the two events just mentioned cannot occur simultaneously, meaning that an ace cannot be a figure. Two events are, however, exhaustive or jointly if at least one of them must occur at a given trial. By extracting a card from a deck of 52, the following two events are exhaustive:

E1 = One face card comes out.

E2 = One number card comes out.

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

Let us now deal with the case of joint probability, both independent and dependent. Given two events, A and B, if the two events are independent (I mean the occurrence of one does not affect the probability of the other), the joint probability of the event is equal to the product of the probabilities of A and B:

$$P(A \cap B) = P(A) \times P(B)$$

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

Let's take an example. We have two decks of 52 cards. By extracting a card from each deck, let's consider the two independent events:

A = The card extracted from the first deck is an ace.

B = The card extracted from the second deck is a clubs card.

What is the probability that both of them occur?

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

$P(A) = 4/52$

$P(B) = 13/52$

$P(A \cap B) = 4/52 \cdot 13/52 = 52 /(52 * 52) = 1/52$

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

If the two events are dependent (that is, the occurrence of one affects the probability of the other), then the same rule may apply, provided P(B|A) is the probability of event A given that event B has occurred. This condition introduces conditional probability, which we are going to dive into:

$$P(A \cap B) = P(A) \times P(B|A)$$

2. Probabilistic classification algorithms - Naive Bayes

Basic concepts of probability

A bag contains 2 white balls and 3 red balls. Two balls are pulled out from the bag in two successive extractions without reintroducing the first ball pulled out into the bag.

Calculate the probability that the two balls extracted are both white:

The probability that the first ball is white is 2/5.

The probability that the second ball is white, provided that the first ball is white, is 1/4.

The probability of having two white balls is….

2. Probabilistic classification algorithms - Naive Bayes

The probability that event A occurs, calculated on the condition that event B occurred, is called conditional probability and is indicated by the symbol P(A | B). It is calculated using the following formula:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Conditional probability usually applies when A depends on B, that is, events are dependent on each other. In the case where A and B are independent, the formula becomes:

$$P(A|B) = P(A)$$

2. Probabilistic classification algorithms - Naive Bayes

Let's take an example. What is the probability that by extracting two cards from a deck of 52, the second one is a diamond? Note the information that the first was a diamond too.

$$P(diamonds \cap diamonds) = 13/52 \cdot 12/51$$

$$P(diamonds \mid diamonds) = (13/52 \cdot 12/51) / 13/52 = 12/51$$

2. Probabilistic classification algorithms - Naive Bayes

Classifying with Naive Bayes

Let A and B be two dependent events. Previously, we said that the joint probability between the two events is calculated using the following formula:

$$P(A \cap B) = P(A) \times P(B|A)$$

Or, similarly, using the following formula:

$$P(A \cap B) = P(B) \times P(A|B)$$

2. Probabilistic classification algorithms - Naive Bayes

$$P(A \cap B) = P(A) \times P(B|A) \qquad P(A \cap B) = P(B) \times P(A|B)$$

By analyzing the two proposed formulas, it is clear that they have the first equal member. This implies that even the second members are equal, so we can write:

$$P(A) \times P(B|A) = P(B) \times P(A|B)$$

By solving these equations for conditional probability, we get:

$$P(B|A) = \frac{P(B) \times P(A|B)}{P(A)} \qquad P(A|B) = \frac{P(A) \times P(B|A)}{P(B)}$$

2. Probabilistic classification algorithms - Naive Bayes

Let's take an example. Suppose you are given two coins. The first coin is fair (heads and tails) and the second coin is wrong (heads on both sides).You randomly choose a coin and toss it, getting heads as a result. What is the likelihood of it being the second coin (wrong coin)?

Let's start by distinguishing the various events that come into play. Let's identifythese events:

A: The first coin was chosen

B: The second coin was chosen

C: After the toss comes a head

2. Probabilistic classification algorithms - Naive Bayes

According to Bayes' theorem, we can write:

$$P(B|C) = \frac{P(B) \times P(C|B)}{P(C)}$$

Now calculate the three probabilities that appear in the previous equation. Remember that P(B|C) is called posterior probability and that is what we want to calculate. P(B) is called prior probability, linked to the second event (B), and is equal to 1/2, since we have two possible choices (two coins are available).

$$P(B) = \frac{1}{2}$$

2. Probabilistic classification algorithms - Naive Bayes

P(C|B) is called likelihood and is equal to 1, as it gives the chances of heads knowing that you have chosen the second coin (which has two heads and so is a certain event). Therefore:

$$P(C|B) = 1$$

Finally, P(C) is called marginal likelihood and is equal to ¾, as the coins have 4 faces (possible cases) of which three have heads (favorable cases).

$$P(C) = \frac{3}{4}$$

2. Probabilistic classification algorithms - Naive Bayes

At this point, we can enter the calculated probabilities in the Bayes formula to get the result:

$$P(B|C) = \frac{P(B) \times P(C|B)}{P(C)} = \frac{\frac{1}{2} \times 1}{\frac{3}{4}} = \frac{2}{3}$$

3. Bayesian methodologies in MATLAB

As anticipated at the beginning of the topic, the Bayesian classifier algorithm assumes that the effect of an event on a given class is independent of the values of other events. This assumption, called the conditional independence of classes, is intended to simplify calculations, and for this reason the algorithm is named naive.

3. Bayesian methodologies in MATLAB

In MATLAB, the Naive Bayes classifier works with data in two steps:

First step of training: Using a set of data, preventively classified, the algorithm estimates the parameters of a probability distribution according to the assumption that predictors are conditionally independent given the class.

Second step of prediction: For a new set of data, the posterior probability of that sample belonging to each class is computed. In this way, new data is classified according to the largest posterior probability.

3. Bayesian methodologies in MATLAB

In particular, we will try to classify the types of iris based on the characteristics of its petals, this time using a Bayesian classifier.

To train a Naive Bayes classifier, we can use the fitcnb() function; it returns a multiclass Naive Bayes model trained by the predictors provided.

In this case, the predictors are the petal length and petal width, while the classes are setosa, versicolor, and virginica.

3. Bayesian methodologies in MATLAB

To import such data, simply type:

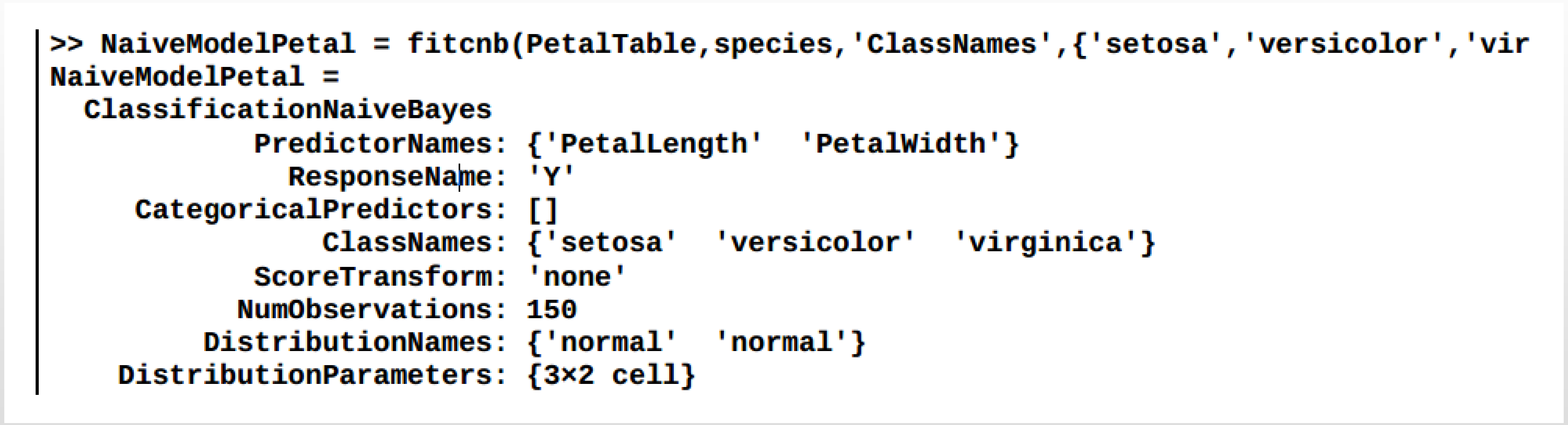
```
>> load fisheriris
```

First, we extract the third and fourth columns of the meas variable, whichcorrespond to petal length and petal width respectively. Then, we create a table with these variables to include in the model the names of the predictors:

```
>> PetalLength = meas(:,3);
>> PetalWidth = meas(:,4);
>> PetalTable = table(PetalLength,PetalWidth);
```

3. Bayesian methodologies in MATLAB

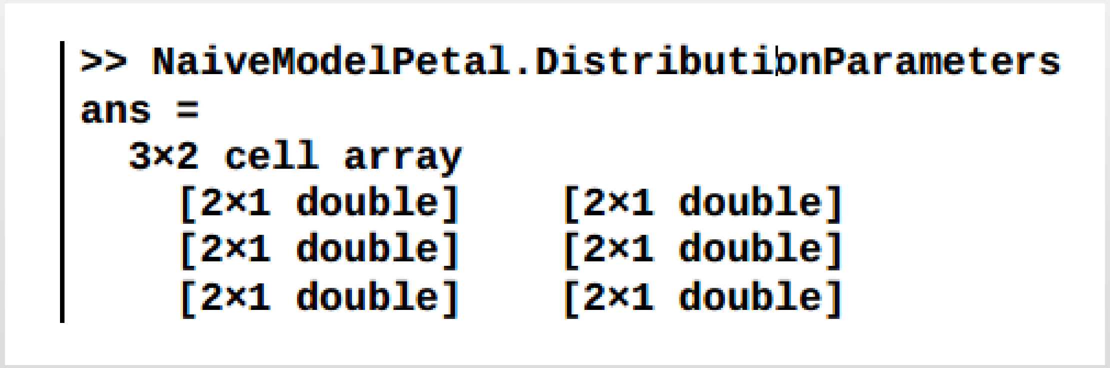
To train a Naive Bayes classifier, type:



```
>> NaiveModelPetal = fitcnb(PetalTable,species,'ClassNames',{'setosa','versicolor','vir
NaiveModelPetal =
  ClassificationNaiveBayes
            PredictorNames: {'PetalLength'  'PetalWidth'}
              ResponseName: 'Y'
     CategoricalPredictors: []
                ClassNames: {'setosa'  'versicolor'  'virginica'}
            ScoreTransform: 'none'
           NumObservations: 150
         DistributionNames: {'normal'  'normal'}
    DistributionParameters: {3×2 cell}
```

3. Bayesian methodologies in MATLAB

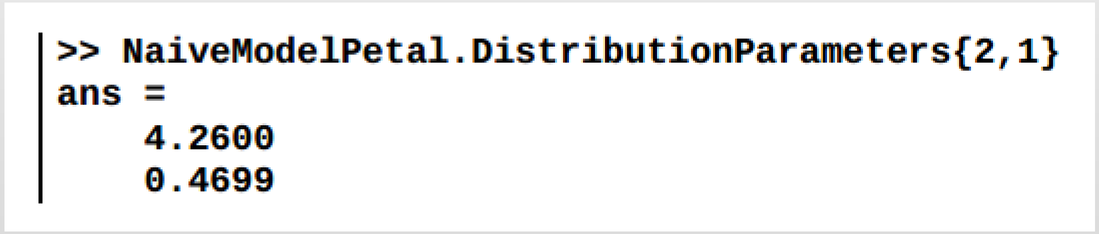
A ClassificationNaiveBayes object is created with several properties and methods. To access one of these, we can use dot notation; for example, to display the mean and standard deviation of a particular Gaussian fit, type this:



```
>> NaiveModelPetal.DistributionParameters
ans =
  3×2 cell array
    [2×1 double]    [2×1 double]
    [2×1 double]    [2×1 double]
    [2×1 double]    [2×1 double]
```

3. Bayesian methodologies in MATLAB

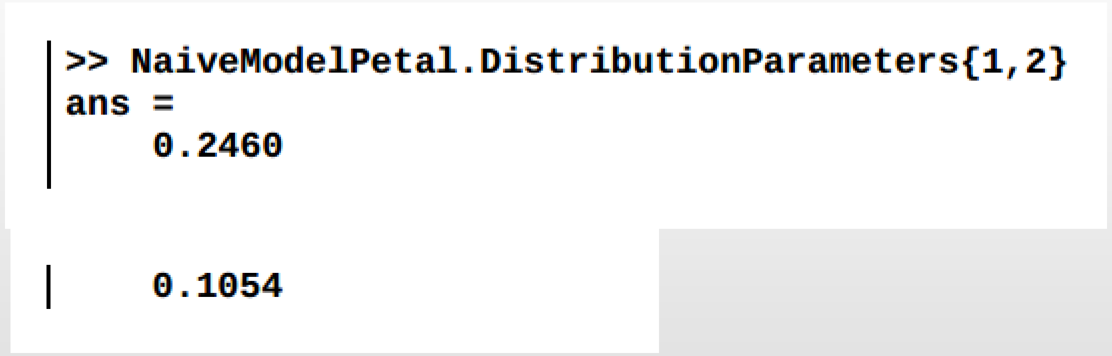
This command returns a 3×2 cell array that contains the mean and standard deviation for each species and each feature (petal length and petal width). Each row contains species data (setosa, versicolor, and virginica respectively), and each column contains feature data (petal length and petal width). Thus, to access the mean and standard deviation of the versicolor species related to the petal length, type:



```
>> NaiveModelPetal.DistributionParameters{2,1}
ans =
    4.2600
    0.4699
```

3. Bayesian methodologies in MATLAB

To access the mean and standard deviation of the setosa species related to the petal width, type this:



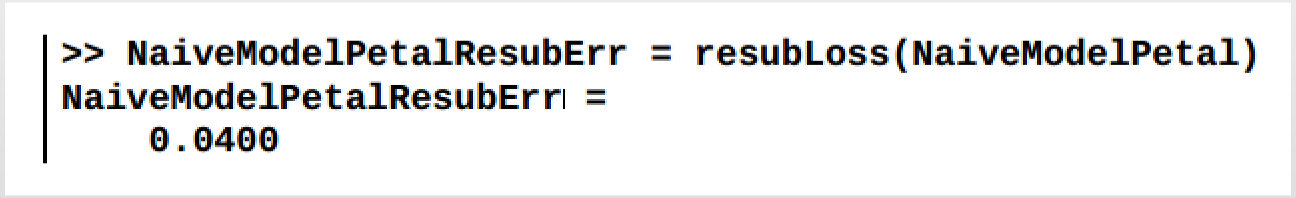
```
>> NaiveModelPetal.DistributionParameters{1,2}
ans =
    0.2460

    0.1054
```

The first value is the mean; the second is the standard deviation.

3. Bayesian methodologies in MATLAB

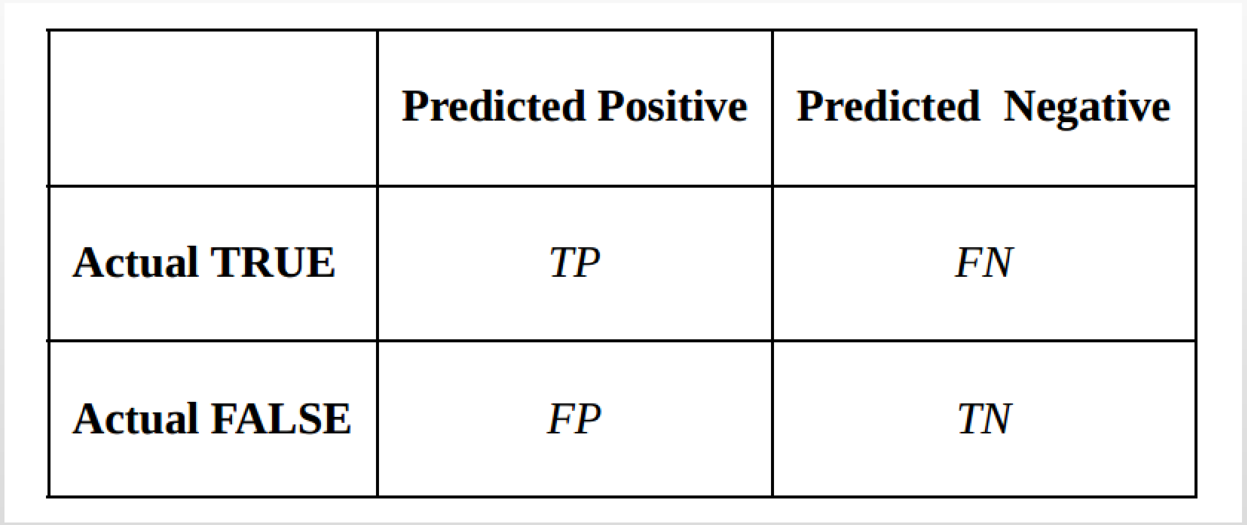
To test the performance of the model, compute the resubstitution error, which is the misclassification error or the proportion of misclassified observations on the training set. The resubstitution error rate indicates only how good/bad our results are; it provides us with some knowledge about the algorithm used (usually, it indicates the performance measure):



```
>> NaiveModelPetalResubErr = resubLoss(NaiveModelPetal)
NaiveModelPetalResubErr =
    0.0400
```

3. Bayesian methodologies in MATLAB

To understand how these errors are distributed, we can compute the confusion matrix.



	Predicted Positive	Predicted Negative
Actual TRUE	*TP*	*FN*
Actual FALSE	*FP*	*TN*

3. Bayesian methodologies in MATLAB

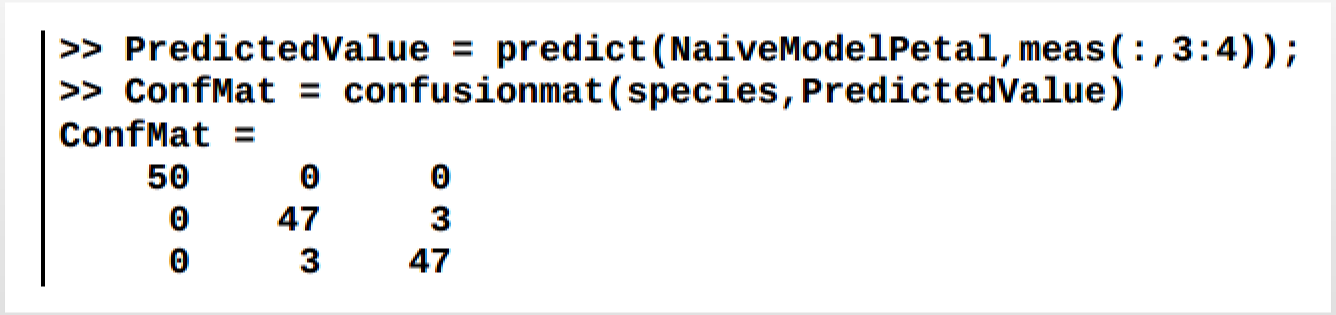
The entries in the confusion matrix have the following meanings:

TP is the number of correct predictions that an instance is positive
FN is the number of incorrect predictions that an instance is negative
FP is the number of incorrect predictions that an instance is positive
TN is the number of correct predictions that an instance is negative

It is clear that the values on the main diagonal represent the correct predictions; all the others are errors.

3. Bayesian methodologies in MATLAB

In MATLAB, to compute a confusion matrix, we can use the confusionmat() function; it returns the confusion matrix determined by the known and predicted groups provided. Before using the function, we collect the model predictions for available data, and then we compute the confusion matrix:



```
>> PredictedValue = predict(NaiveModelPetal,meas(:,3:4));
>> ConfMat = confusionmat(species,PredictedValue)
ConfMat =
    50     0     0
     0    47     3
     0     3    47
```

As expected, there are only six errors (four percent of the available values) and they refer to the two versicolor and virginica species.

4. Find similarities using nearest neighbor classifiers

Among the different types of existing classifiers, we also find the nearest neighbor, which identifies the class of belonging to a tested sample based on the distance of this from stored and classified objects. In most cases, the distance is defined as Euclidean distance between two points, calculated according to the following formula:

$$Distance = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

4. Find similarities using nearest neighbor classifiers

An object is assigned to the class based on the majority vote of its neighbors, the most common among its KNN is chosen (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. This type of classifier, however, is not very sensitive, because it is subject to frequent errors (since it ranks based on only one stored object).

4. Find similarities using nearest neighbor classifiers

Therefore, you usually choose a larger K (between 1 and 10) and assignment of an object to a given class is based on a simple majority criterion. Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise, as it reduces the overall noise, but there is no guarantee. In most cases, an odd K is chosen to avoid parity situations; otherwise the classification may also take into account the distances of the K objects from the sample to be graded.

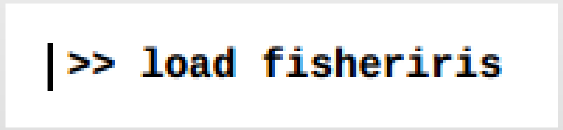
4. Find similarities using nearest neighbor classifiers

A great advantage of this method is to be able to classify objects whose classes are not linearly separable. It is also a very stable classifier, as small perturbations in training data do not significantly affect the results obtained.

The most obvious disadvantage, however, is that it does not provide a true mathematical model, but each new classification must be made by adding the new datum to the initial set and repeating the calculation procedure for the selected K value. It also needs a fairly large amount of data to make realistic predictions and is sensitive to the noise of the data being analyzed.

4. Find similarities using nearest neighbor classifiers

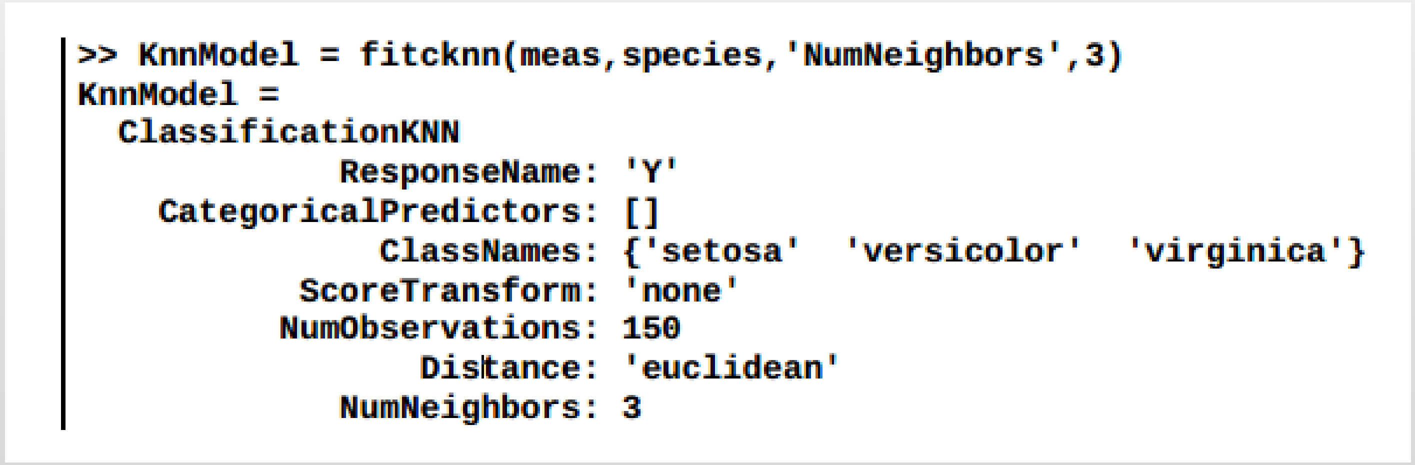
In MATLAB, the KNN classifier is constructed through the fitcknn() function, which returns a KNN classification model based on the predictors and the response provided. Again, we will use the fisheriris dataset to compare the different classification algorithms proposed in this chapter:



```
>> load fisheriris
```

4. Find similarities using nearest neighbor classifiers

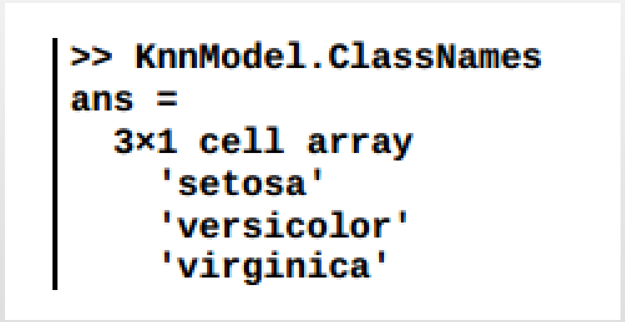
This command creates two variables: meas and species. The first contains the data for the length and width of the sepal and petal (150x4 double). The second covers the classification (150x1 cell). To construct a KNN classifier for this data with k, the number of nearest neighbors in the predictors, equal to 3, simply type:



```
>> KnnModel = fitcknn(meas,species,'NumNeighbors',3)
KnnModel =
  ClassificationKNN
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: {'setosa'  'versicolor'  'virginica'}
           ScoreTransform: 'none'
          NumObservations: 150
                 Distance: 'euclidean'
             NumNeighbors: 3
```

4. Find similarities using nearest neighbor classifiers

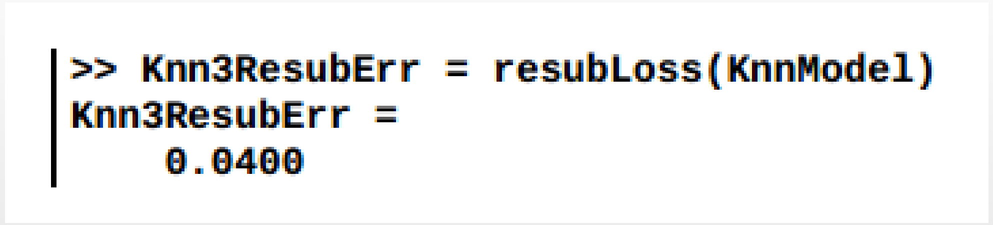
To access the properties of the model just created, use dot notation. For example, to access the name of the response variable, simply type:



```
>> KnnModel.ClassNames
ans =
  3×1 cell array
    'setosa'
    'versicolor'
    'virginica'
```

4. Find similarities using nearest neighbor classifiers

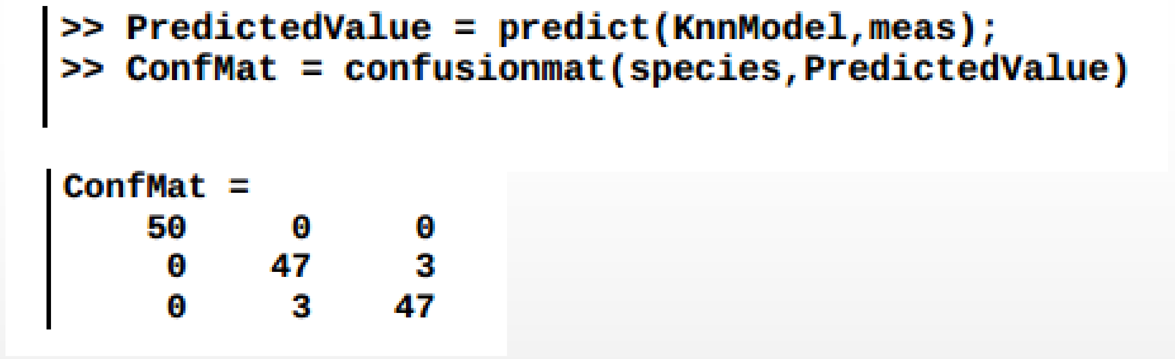
To test the performances of the model, compute the resubstitution error.



```
>> Knn3ResubErr = resubLoss(KnnModel)
Knn3ResubErr =
    0.0400
```

The result indicates that 4 percent of the observations are misclassified by the KNN algorithm. To understand how these errors are distributed, we can compute the confusion matrix.

4. Find similarities using nearest neighbor classifiers



```
>> PredictedValue = predict(KnnModel,meas);
>> ConfMat = confusionmat(species,PredictedValue)
```

```
ConfMat =
    50     0     0
     0    47     3
     0     3    47
```

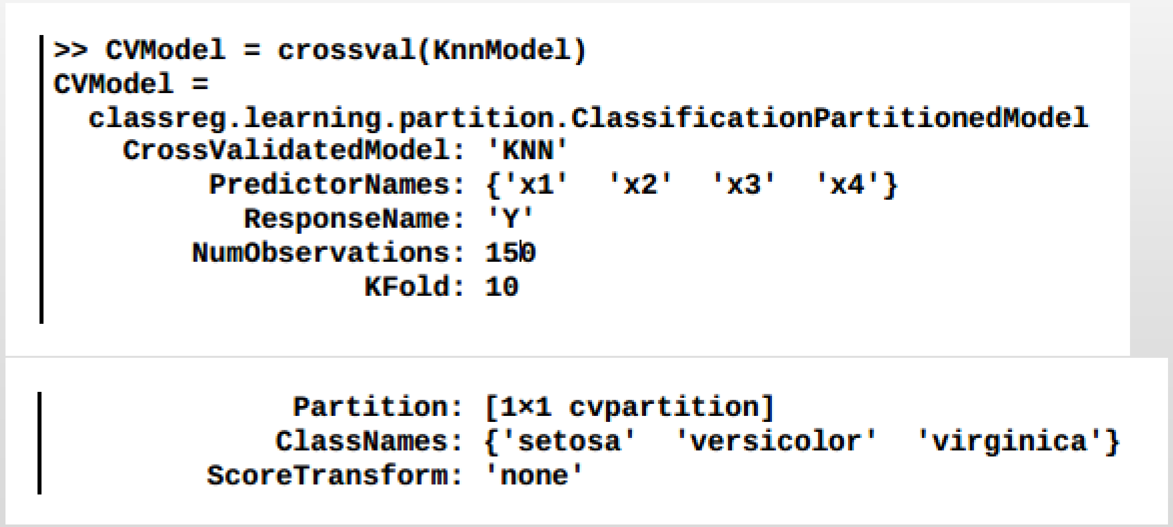
As expected, there are only seven errors and they refer to the two versicolor and virginica species. Previously, we said that the simple calculation of the classification error does not allow us to understand the type of errors committed by our system. In that case, the confusion matrix comes to our rescue, because it allows you to evaluate how errors and correct decisions made by our classifier are distributed.

4. Find similarities using nearest neighbor classifiers

Since performance estimates depend on the data used, simply dividing random data between training and test sets does not guarantee that results are statistically significant. The repetition of the evaluation on different random divisions and the calculation of performance in terms of the mean and standard deviation of individual ratings make it possible to have a more reliable estimate. However, repetition of evaluations across random divisions may also prevent the most complex data being graded when evaluating (or training). The solution to these problems is offered by cross-validation.

4. Find similarities using nearest neighbor classifiers

In MATLAB, cross-validation is performed by the crossval() function, which creates a partitioned model from a fitted KNN classification model. By default, this function uses 10-fold cross-validation on the training data to create the model:

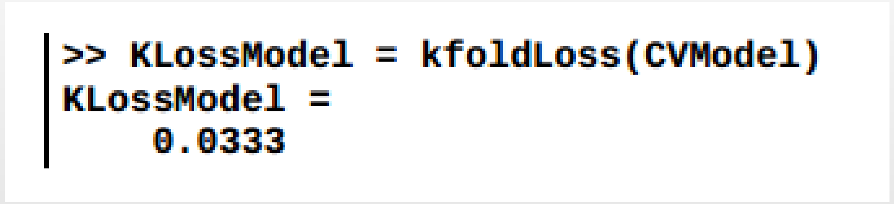


```
>> CVModel = crossval(KnnModel)
CVModel =
  classreg.learning.partition.ClassificationPartitionedModel
    CrossValidatedModel: 'KNN'
         PredictorNames: {'x1'  'x2'  'x3'  'x4'}
           ResponseName: 'Y'
        NumObservations: 150
                  KFold: 10
```

```
              Partition: [1×1 cvpartition]
             ClassNames: {'setosa'  'versicolor'  'virginica'}
         ScoreTransform: 'none'
```

4. Find similarities using nearest neighbor classifiers

Now, we will visualize the cross-validation loss, which is the average loss of each cross-validation model when prediction is executed on data that is not used for training.

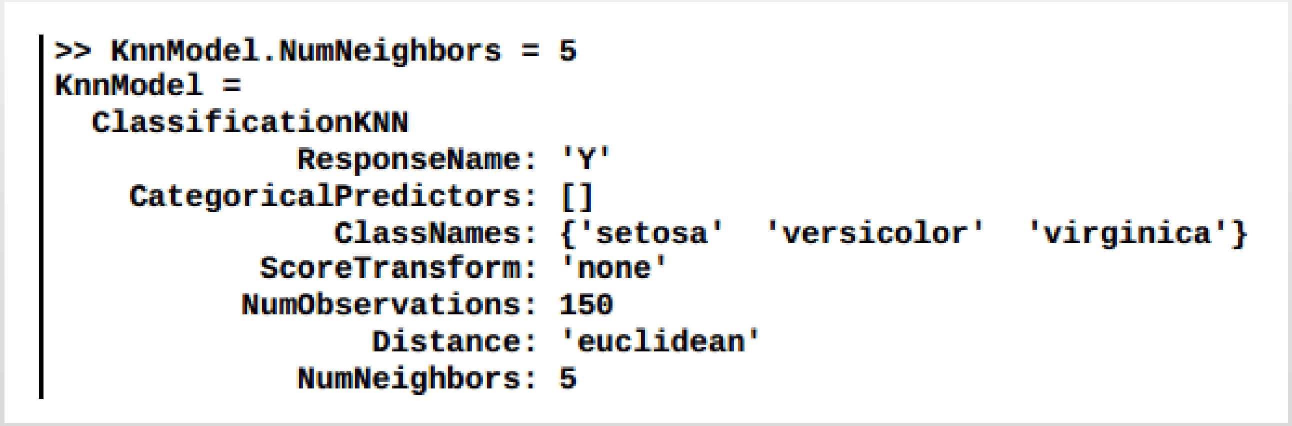


```
>> KLossModel = kfoldLoss(CVModel)
KLossModel =
    0.0333
```

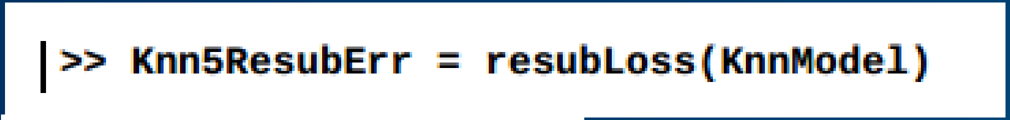
In this case, the cross-validated classification accuracy is very close to the resubstitution accuracy.

4. Find similarities using nearest neighbor classifiers

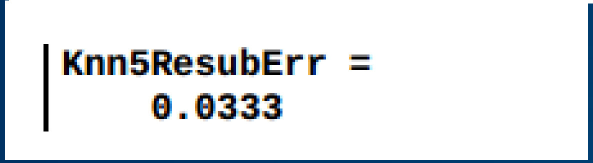
Previously, we said that choosing the optimal value for K is very important in order to create the best classification model. So, we try to change the value of k and see what happens. To change the k value simply modifies the NumNeighbors property of the model. Let's see what happens by fixing k = 5:



```
>> KnnModel.NumNeighbors = 5
KnnModel =
  ClassificationKNN
              ResponseName: 'Y'
     CategoricalPredictors: []
                ClassNames: {'setosa'  'versicolor'  'virginica'}
            ScoreTransform: 'none'
           NumObservations: 150
                  Distance: 'euclidean'
              NumNeighbors: 5
```

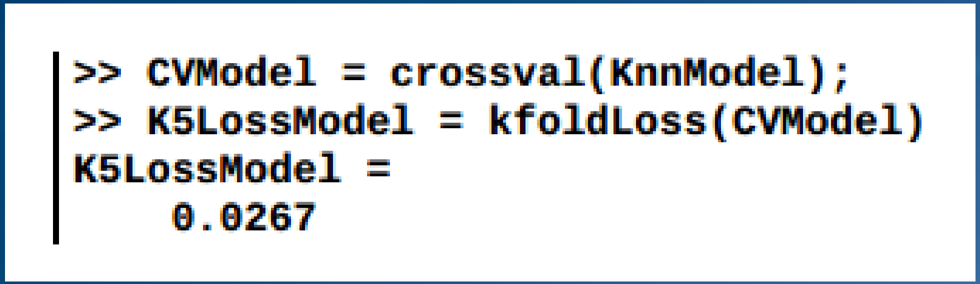


```
>> Knn5ResubErr = resubLoss(KnnModel)
```



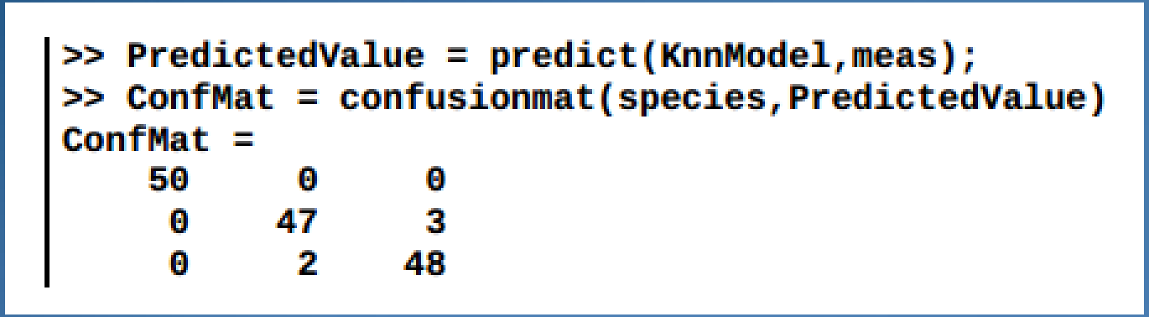
```
Knn5ResubErr =
    0.0333
```

We run cross-validation again and display the cross-validation loss.



```
>> CVModel = crossval(KnnModel);
>> K5LossModel = kfoldLoss(CVModel)
K5LossModel =
    0.0267
```

In this regard, we recalculate the matrix of confusion:

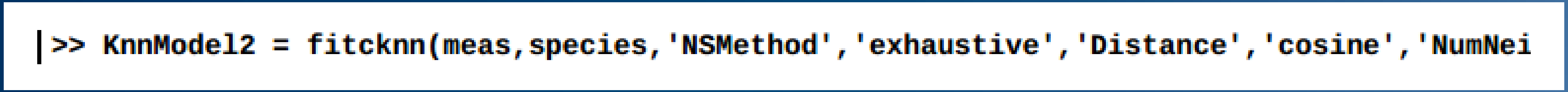


```
>> PredictedValue = predict(KnnModel,meas);
>> ConfMat = confusionmat(species,PredictedValue)
ConfMat =
    50     0     0
     0    47     3
     0     2    48
```

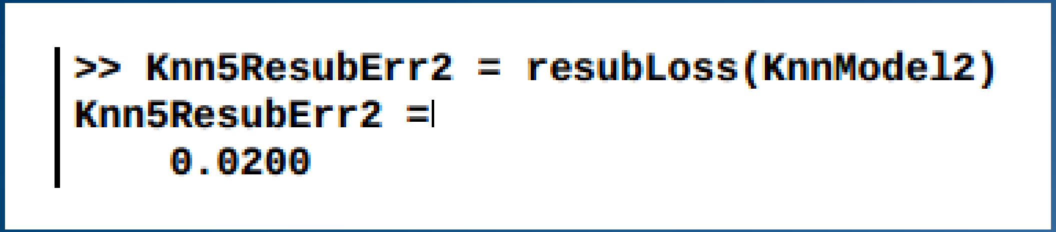
4. Find similarities using nearest neighbor classifiers

We continue in our work to improve model performance. At the beginning of the section, we saw that the fitcknn() function uses Euclidean distance by default to evaluate the nearest neighbor.

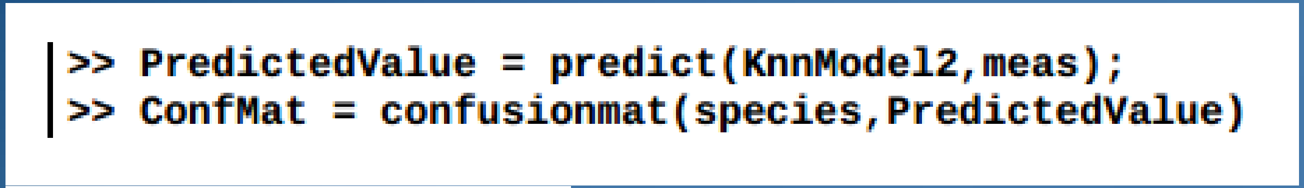
To set the distance metric, we need to specify the comma-separated pair consisting of Distance and a valid distance metric name or function handle. For example, we now modify the model to use cosine distance instead of the default. To use cosine distance, you must recreate the model using the exhaustive search method:



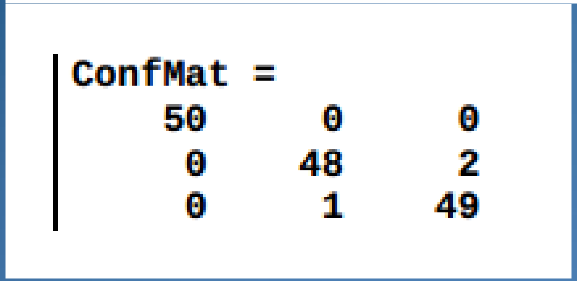
```
>> KnnModel2 = fitcknn(meas,species,'NSMethod','exhaustive','Distance','cosine','NumNei
```



```
>> Knn5ResubErr2 = resubLoss(KnnModel2)
Knn5ResubErr2 =
    0.0200
```



```
>> PredictedValue = predict(KnnModel2,meas);
>> ConfMat = confusionmat(species,PredictedValue)
```



```
ConfMat =
    50     0     0
     0    48     2
     0     1    49
```