Classification of iris flowers

Abstract

The aim is to classify iris flowers among three species (setosa, versicolor, or virginica) from measurements of sepals and petals' length and width. The iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The central goal here is to design a model that makes useful classifications for new flowers or, in other words, one which exhibits good generalization.

Theory

This is a classification project, since the variable to be predicted is categorical (setosa, versicolor, or virginica). The goal here is to model the probabilities of class membership, conditioned on the flower features.

The data source is the file Iris.csv. It contains the data for this example in comma-separated values (CSV) format. The number of columns is 6, and the number of rows is 150.

The variables are:

- Id: Id number of all records.
- SepalLengthCm: Sepal length, in centimeters, used as input.
- SepalWidthCm: Sepal width, in centimeters, used as input.
- PetalLengthCm: Petal length, in centimeters, used as input.
- PetalWidthCm: Petal width, in centimeters, used as input.
- Species: Iris Setosa, Versicolor, or Virginica, used as the target.

KNN Classifier

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining.

KNN can be used for both classification and regression predictive problems. KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consisting of training observations (x,y) and would like to capture the relationship between x and y . More formally, our goal is to learn a function h:X→Y so that given an unseen observation x, h(x) can confidently predict the corresponding output y.

KNN can be summarized as

1. Computes the distance between the new data point with every training example.
2. For computing the distance measures such as Euclidean distance, Hamming distance or Manhattan distance will be used.
3. Model picks K entries in the database which are closest to the new data point.
4. Then it does the majority vote i.e the most common class/label among those K entries will be the class of the new data point.

As we can see labels are categorical. KNeighborsClassifier does not accept string labels. We need to use LabelEncoder to transform them into numbers. Iris-setosa correspond to 0, Iris-versicolor correspond to 1 and Iris-virginica correspond to 2.

Let's split dataset into training set and test set, to check later on whether or not our classifier works correctly. The training and test split ratio is 70:30 where, 70% of data is stored in training variables X_train and X_test whereas, remaining 30% of data is stored in testing variables y_train and y_test.

Accuracy

The accuracy obtained using KNN Classifier is 97.78%. The optimal number of neighbors required is 7.