**ALGORITHM 2**

**IMPLEMENTATION OF KNN ALGORITHM USING IRIS DATASET**

**CHAPTER 1**

**CASE STUDY DESCRIPTION**

Implementation of KNN Algorithm using Iris dataset

**1.1 INTRODUCTION**

K Nearest Neighbour Classification Algorithm popularly known by the name KNN classifiers. We will mainly focus on learning to build your first KNN model.

Classification Machine Learning is a technique of learning where a particular instance is mapped against one of the many labels. The labels are prespecified to train your model. The machine learns the pattern from the data in such a way that the learned representation successfully maps the original dimension to the suggested label/class without any more intervention from a human expert.

**1.2 HOW DOES KNN WORKS**

The KNN algorithm is belongs to the family of instance-based, competitive learning and lazy learning algorithms.

Instance-based algorithms are those algorithms that model the problem using data instances (or rows) in order to make predictive decisions. The KNN algorithm is an extreme form of instance-based methods because all training observations are retained as part of the model. KNN is powerful because it does not assume anything about the data, other than a distance measure can be calculated consistently between any two instances. As such, it is called non-parametric or non-linear as it does not assume a functional form.

**CHAPTER 2**

**ALGORITHM INTRODUCTION**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

**2.1 KNN WORKING**

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of K number of neighbors
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.

**CHAPTER 3**

**3.1 IMPLEMENTATION**

Implementing KNN Algorithm.

**The Dataset**

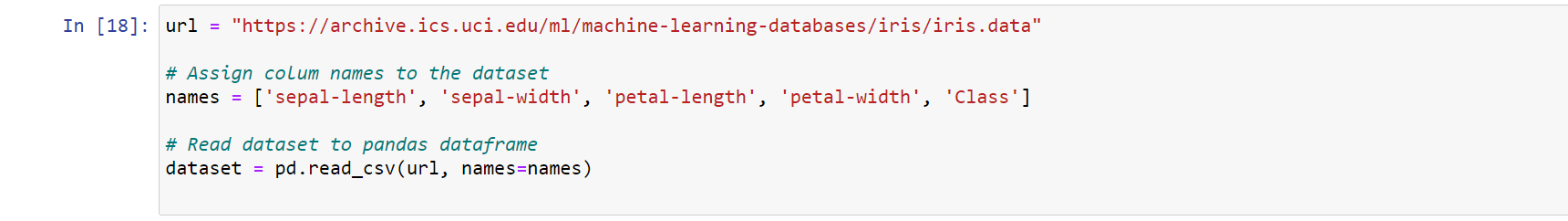
We are going to use the famous iris data set for our KNN example. The dataset consists of four attributes: sepal-width, sepal-length, petal-width and petal-length. These are the attributes of specific types of iris plant. The task is to predict the class to which these plants belong. There are three classes in the dataset: Iris-setosa, Iris-versicolor and Iris-virginica.

1. **Importing Libraries**

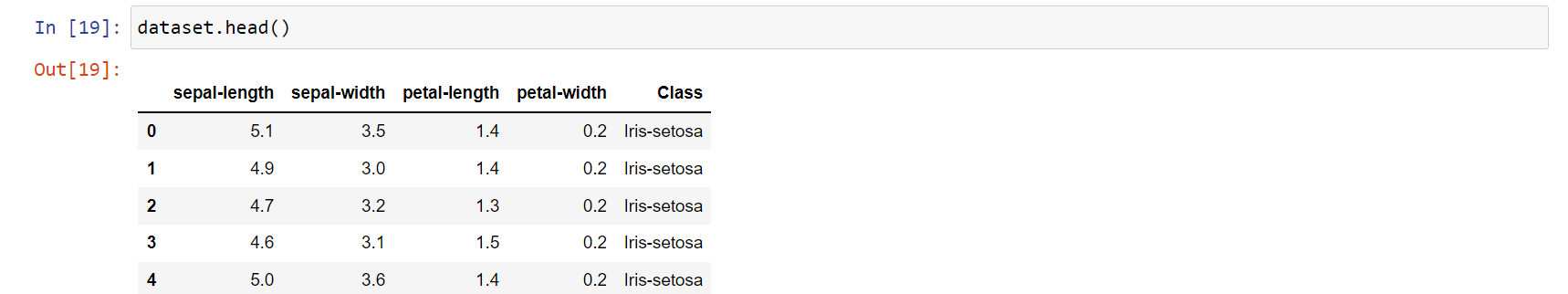


1. **Importing the Dataset**

To import the dataset and load it into our pandas dataframe, execute the following code:

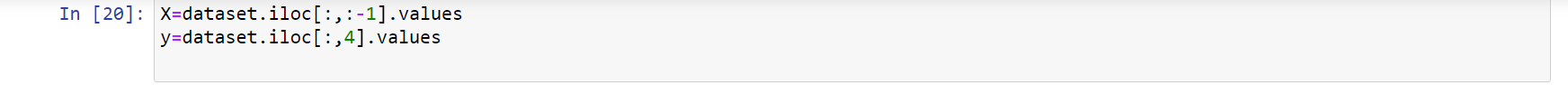
****

To see what the dataset actually looks like, execute the following command:



1. **Preprocessing**

The next step is to split our dataset into its attributes and labels. To do so, use the following code:

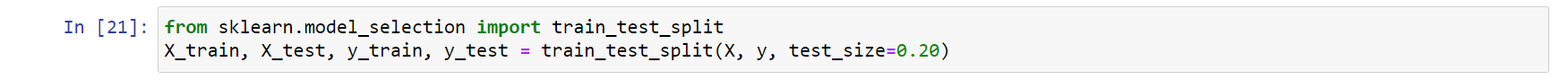


The X variable contains the first four columns of the dataset (i.e. attributes) while y contains the labels.

1. **Train Test Split**

To avoid over-fitting, we will divide our dataset into training and test splits, which gives us a better idea as to how our algorithm performed during the testing phase. This way our algorithm is tested on un-seen data, as it would be in a production application.

To create training and test splits, execute the following script:



The above script splits the dataset into 80% train data and 20% test data. This means that out of total 150 records, the training set will contain 120 records and the test set contains 30 of those records.

1. **Feature Scaling**

Before making any actual predictions, it is always a good practice to scale the features so that all of them can be uniformly evaluated.

The gradient descent algorithm (which is used in neural network training and other machine learning algorithms) also converges faster with normalized features.

The following script performs feature scaling:



1. **Training and Predictions**

It is extremely straight forward to train the KNN algorithm and make predictions with it, especially when using Scikit-Learn.



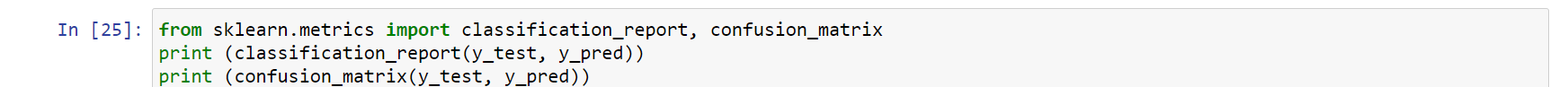
The first step is to import the KNeighborsClassifier class from the sklearn.neighbors library. In the second line, this class is initialized with one parameter, i.e. n\_neigbours. This is basically the value for the K. There is no ideal value for K and it is selected after testing and evaluation, however to start out, 5 seems to be the most commonly used value for KNN algorithm.

The final step is to make predictions on our test data. To do so, execute the following script:

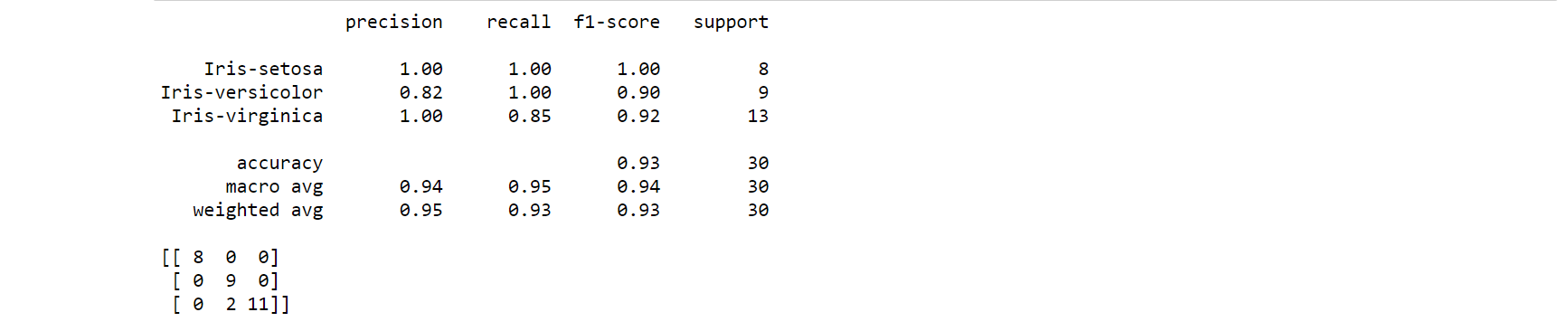


1. **Evaluating the Algorithm**

For evaluating an algorithm, confusion matrix, precision, recall and f1 score are the most commonly used metrics. The confusion\_matrix and classification\_report methods of the sklearn.metrics can be used to calculate these metrics. Take a look at the following script:



The output of the above script looks like this:



The results show that our KNN algorithm was able to classify all the 30 records in the test set with 100% accuracy, which is excellent. Although the algorithm performed very well with this dataset, don't expect the same results with all applications. As noted earlier, KNN doesn't always perform as well with high-dimensionality or categorical features.

1. **Comparing Error Rate with the K Value**

In the training and prediction section we said that there is no way to know beforehand which value of K that yields the best results in the first go. We randomly chose 5 as the K value and it just happen to result in 100% accuracy.

One way to help you find the best value of K is to plot the graph of K value and the corresponding error rate for the dataset.

In this section, we will plot the mean error for the predicted values of test set for all the K values between 1 and 40.

To do so, let's first calculate the mean of error for all the predicted values where K ranges from 1 and 40. Execute the following script:



The above script executes a loop from 1 to 40. In each iteration the mean error for predicted values of test set is calculated and the result is appended to the error list.

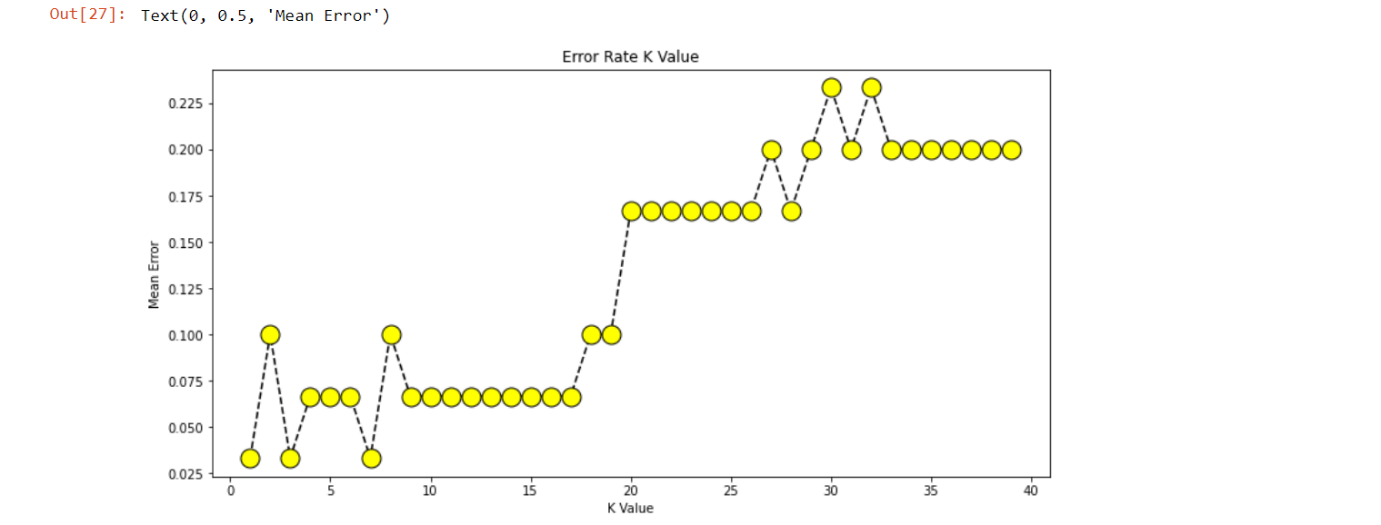
The next step is to plot the error values against K values. Execute the following script to create the plot:



**CHAPTER 4**

**4.1 OUTCOME**

The output graph looks like this:



From the output we can see that the mean error is zero when the value of the K is between 5 and 18. I would advise you to play around with the value of K to see how it impacts the accuracy of the predictions.