Supervised machine learning on Iris flower dataset using KNN algorithm

https://github.com/jecece1/Supervised-machine-learning-on-Iris-flower-dataset-using-KNN-algorithm.git

Praveen Elango – 2015115099 (IT)

Problem definition

To design and implement the Identification of Iris Flower species using machine learning using Python and the tool Scikit-Learn.

Project domain

The domain of our project is **Machine Learning**.

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$\mathbf{Y = f(X)}$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems.

- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.
  Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

- **Linear regression** for regression problems.
- **Random forest** for classification and regression problems.

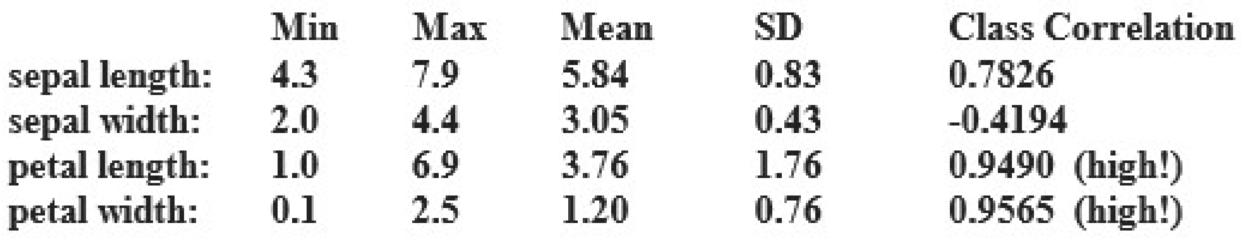
- **Support vector machines** for classification problems.
- **k-Nearest Neighbors** for classification problems based on similarity measures.

Dataset used

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis. It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris Flower of three related species. Two of the three species were collected in Gaspe Peninsula all from the same pasture and picked on the same day and measured at the same time by the same person with same apparatus.

The data set consists of 50 samples from each of three species of Iris that is : 1) Iris Setosa 2) Iris Virginica 3) Iris Versicolor. Four features were measured from each sample. They are 1) Sepal Length 2) Sepal Width 3) Petal Length 4) Petal Width. All these four parameters are measured in Centimeters. Based on the combination of these four features, the species among three can be predicted from a total of 150 records.

Summary Statistics



	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

Algorithm used

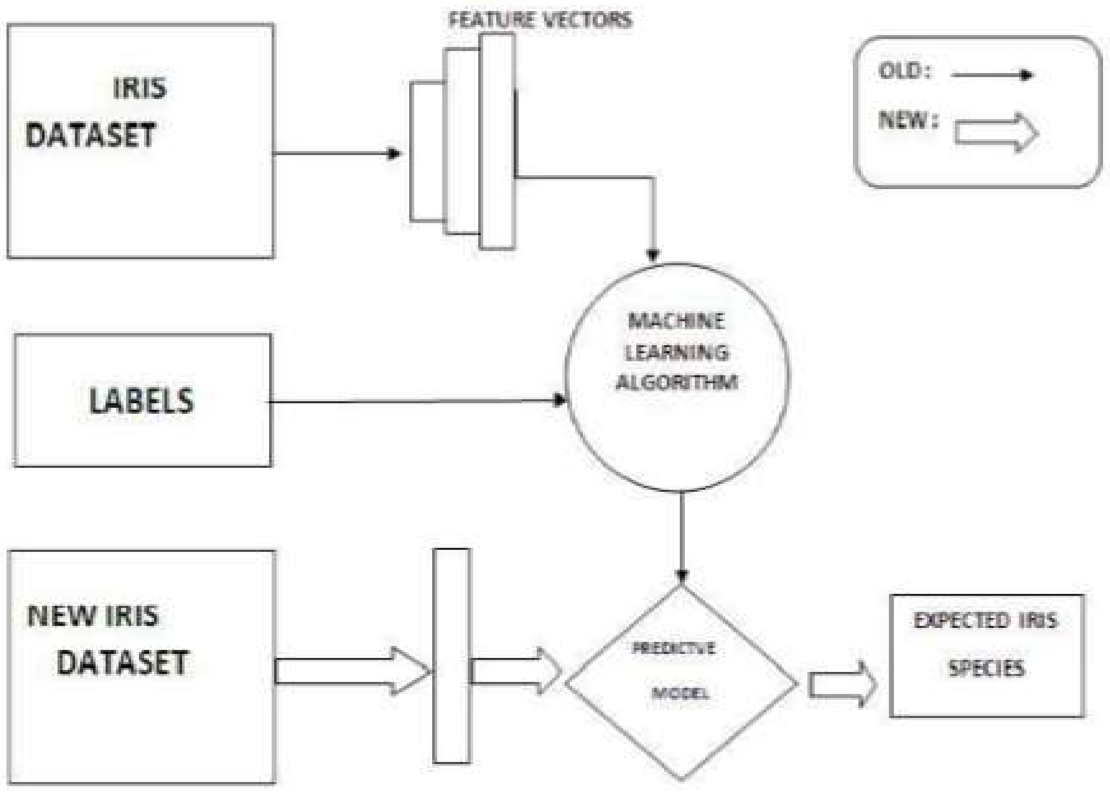
K nearest neighbours is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). It is probably the most simplistic machine learning algorithm because it doesn’t make any mathematical assumptions and doesn’t require heavy machinery. It just requires an understanding of distances between points which are the Euclidian or Manhattan distances. The only assumption for this algorithm is that **the points that are close to one another are similar.** This will allow us to classify any new points. The new point will be classified by its nearest neighbors, or majority of nearest neighbors if there are multiple. k-Nearest Neighbors can be considered a lazy algorithm because there is no learning phrase from this model. It simply memorizes the training data and compares it to our test data.

With this algorithm we commonly measure the Euclidean distance, the ordinary straight line, between two points. The Euclidean distance equation is:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

In this equation **p** and **q** are two points in Euclidean *n*-space. All we have to do is measure this distance of our test point to every training data point and then assign a predicted label to the test, based on the points that are closest to it. This algorithm will give us a relatively high prediction accuracy and the classifier can be imported from the Scikit-Learn library.

Block Diagram



Technology stack

Language : Python and associated libraries

IDE : Jupyter Notebook

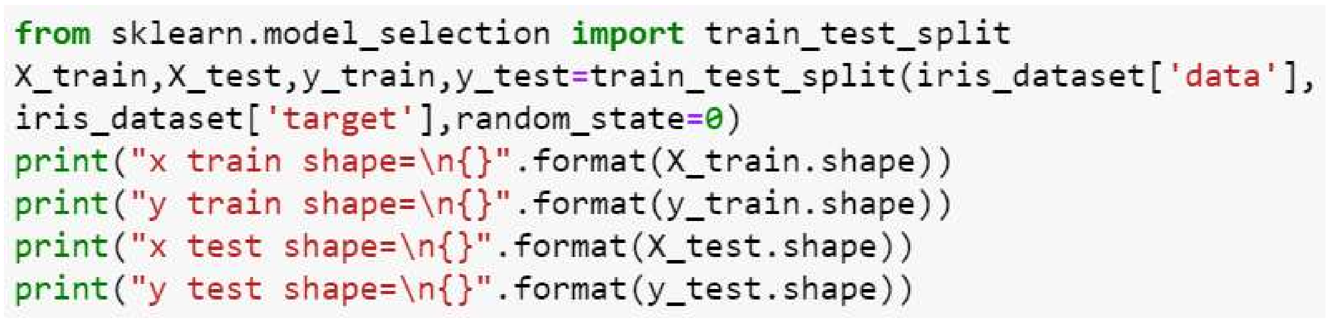
Implementation

i) Loading the salaries data set

First, it’s important to import all the required libraries for our project.

```
#Load libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

Then we have to load the IRIS flowers training data set and assign it to a variable called "dataset". We are using sklearn to load the data. We will use pandas next to explore the data both with descriptive statistics and data visualization.

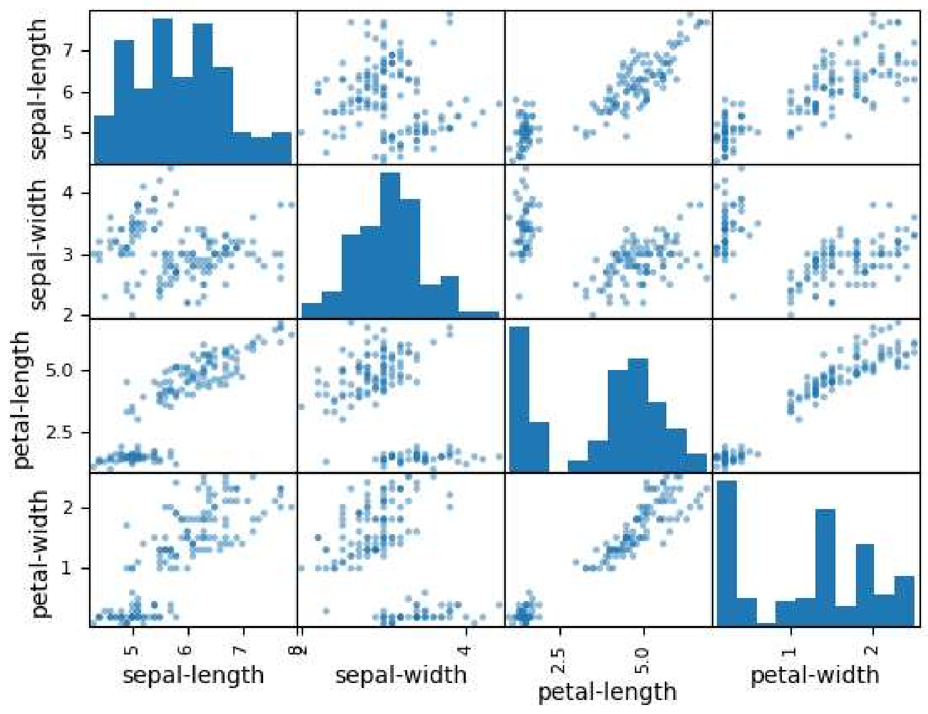


```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(iris_dataset['data'],
iris_dataset['target'],random_state=0)
print("x train shape=\n{}".format(X_train.shape))
print("y train shape=\n{}".format(y_train.shape))
print("x test shape=\n{}".format(X_test.shape))
print("y test shape=\n{}".format(y_test.shape))
```

ii) Analysing the data visually

To understand how each feature for classification of the data, we can build a scatter plot which shows us the correlation with respect to other features. This method helps just to figure out the important features which account the most for the classification in our model.

```
import mglearn
from pandas.plotting import scatter_matrix
iris_dataframe=pd.DataFrame(X_train,columns=iris_dataset.feature_names)
grr=scatter_matrix(iris_dataframe,c=y_train,figsize=(15,15),marker='o',
                   hist_kwds={'bins':20},s=40,alpha=1,cmap=mglearn.cm3)
```

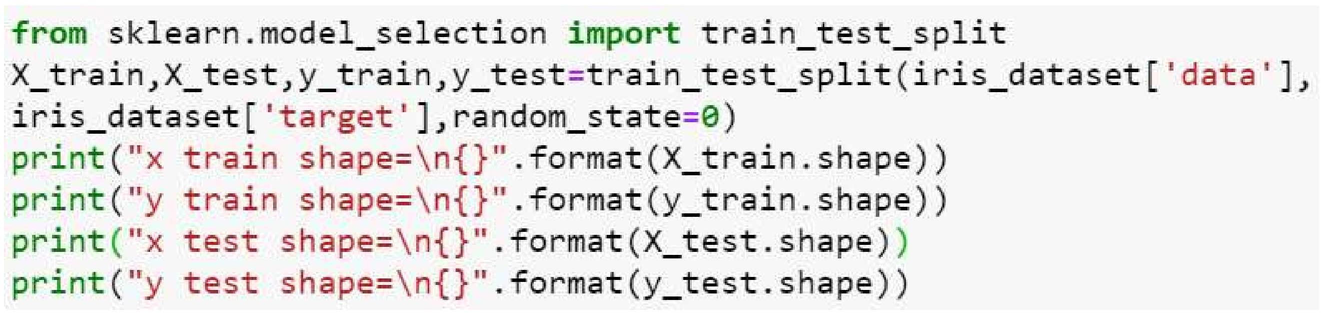


iii) Splitting the Data

In Machine learning we have two kinds of datasets

- **Training dataset** - used to train our model
- **Testing dataset** - used to test if our model is making accurate predictions

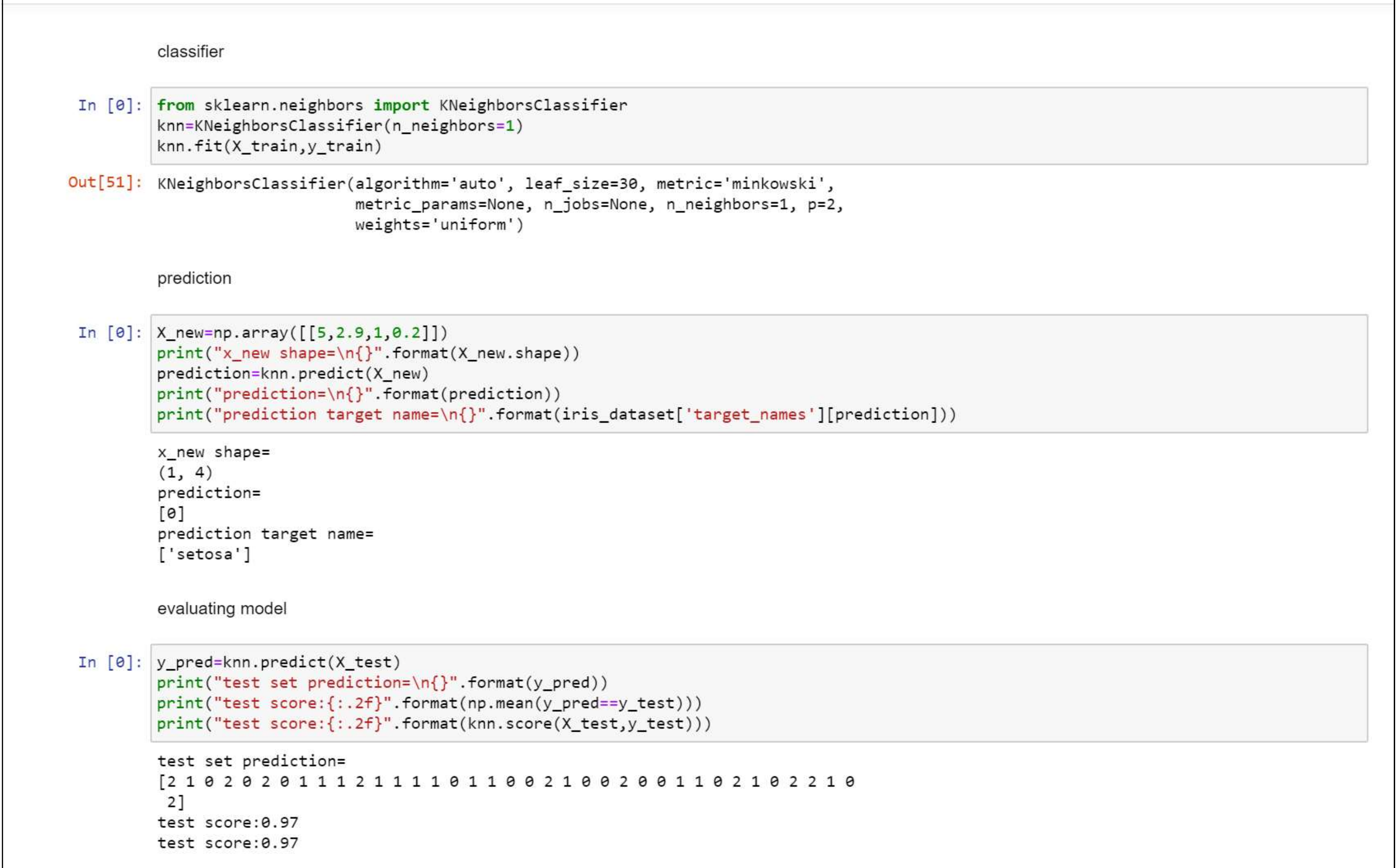
Since our dataset is small (150 records) we will use 120 records for training the model and 30 records to evaluate the model.



```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(iris_dataset['data'],
iris_dataset['target'],random_state=0)
print("x train shape=\n{}".format(X_train.shape))
print("y train shape=\n{}".format(y_train.shape))
print("x test shape=\n{}".format(X_test.shape))
print("y test shape=\n{}".format(y_test.shape))
```

Results

Let’s apply the KNN Algorithm and train the model and predict the species name and accuracy with our testing dataset.



classifier

```
In [0]: from sklearn.neighbors import KNeighborsClassifier
        knn=KNeighborsClassifier(n_neighbors=1)
        knn.fit(X_train,y_train)
```

```
Out[51]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                              weights='uniform')
```

prediction

```
In [0]: X_new=np.array([[5,2.9,1,0.2]])
        print("x_new shape=\n{}".format(X_new.shape))
        prediction=knn.predict(X_new)
        print("prediction=\n{}".format(prediction))
        print("prediction target name=\n{}".format(iris_dataset['target_names'][prediction]))
```

```
x_new shape=
(1, 4)
prediction=
[0]
prediction target name=
['setosa']
```

evaluating model

```
In [0]: y_pred=knn.predict(X_test)
        print("test set prediction=\n{}".format(y_pred))
        print("test score:{:.2f}".format(np.mean(y_pred==y_test)))
        print("test score:{:.2f}".format(knn.score(X_test,y_test)))
```

```
test set prediction=
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
test score:0.97
test score:0.97
```

As we can see, we have successfully imported and applied the KNN classifier to train the model and have predicted the species name for the specified values to an accuracy of 97%.