Classifiction

Write Python code, Jupyter Notebook, that compare between the behavior of three distinct classifiers on your own dataset.

* The classifier’s behavior can be determined by evaluation metrics such as: Classifier’s Accuracy and Precision, Recall and F-measure for each Class in your dataset.

Notes:

* You can use any three classifiers from (KNN, Naïve Bayes, Decision Tree, MLP Classifier)
* Submit your Python code, enrich it with useful comments.
* Report the behavior of the classifiers in Word’s document, or PDF, that describes your experiment.

I select 3 classifiers: KNN, Decision Tree, MLP Classifier.

I will be using the IRIS dataset that's built into scikit learn. Since this dataset is already simplified and has only 4 dimensions (or variables), using the neighbors classifier will be easy. It will be much harder as the number of dimensions increase.

This is used to classify the variety of iris by the length and width of petal and sepal.

1)KNN Classifier

(https://notebook.community/RagsX137/TF\_Tutorial/My%2Bown%2BKNN%2BClassifier)

## - Importing the dataset from sklearn

The Iris Dataset is already loaded in sklearn and more details about iris can be found here.

In [1]:

from sklearn import datasetsiris = datasets.load\_iris()

In [2]:

X = iris.data

# Iris.data contains the features or independent variables.

y = iris.target

# Iris.target contains the labels or the dependent variables.

## - Doing a train-test split by using 50% of the data as our training set

The train-test-splitter found in the cross-valiation (now model selection module) of sklearn is a simple but powerful tool to randomly split the data into train and test datasets.

In [3]:

from sklearn.model\_selection import train\_test\_split

In [4]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = 0.50)

If you have used Machine Learning Classifiers in Python before, remember that there are usually 5 steps involved :

## - Final Steps

#### The classifier is built to utilize the standard pipeline that we use in scikit learn i.e. :

#Call the classifier

In [7]:

my\_classifier = OneNeighborClassifier()

#Fit the model to train it

my\_classifier.fit(X\_train, y\_train)

# Predict the model on the test set

In [8]:

pred = my\_classifier.predict(X\_test)

Check for accuracy between the real values and the predicted values

In [9]:

from sklearn.metrics import accuracy\_scoreprint ('Accuracy of the classifier is', accuracy\_score(y\_test, pred)\*100, '%')

Accuracy of the KNN classifier is 97%

2)Decision Tree -0.96

<https://benalexkeen.com/decision-tree-classifier-in-python-using-scikit-learn/)>

from sklearn import tree

model = tree.DecisionTreeClassifier()

Let’s take a look at our model’s attributes

In [9]:

model

Out[9]:

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=None, splitter='best')

Defining some of the attributes like max\_depth, max\_leaf\_nodes, min\_impurity\_split, and min\_samples\_leaf can help prevent overfitting the model to the training data.

First we fit our model using our training data.

In [10]:

model.fit(X\_train, y\_train)

Out[10]:

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=None, splitter='best')

Then we score the predicted output from model on our test data against our ground truth test data.

Accuracy of the KNN classifier is 96%

3)MLP classifier

<https://www.codersarts.com/forum/machine-learning-tutorial/mlp-classifier-in-python>

Here some important libraries which use to implement MLP Classifier in python

# load libraries

from sklearn import datasets

from sklearn import metrics

from sklearn.neural\_network import MLPClassifier

from sklearn.neural\_network import MLPRegressor

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

import seaborn as sns

**Importing the Dataset**

Here we are using the iris data from sklearn

# load the iris datasets

dataset = datasets.load\_iris()

X = dataset.data; y = dataset.target

**Split data sets**

Now we will split the data using **train\_test\_split**

#split dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

**Fit it into the model**

Now we are ready to fit it into the model

# fit a CART model to the data

model = MLPClassifier()

model.fit(X\_train, y\_train)

print(); print(model)

**Make Prediction**

Now we are predicting the model

# make predictions

expected\_y = y\_test

predicted\_y = model.predict(X\_test)

**Classification report and confusion matrix**

Now, here we will find the result and confusion matrix

# summarize the fit of the model

print(); print(metrics.classification\_report(expected\_y, predicted\_y))

print(); print(metrics.confusion\_matrix(expected\_y, predicted\_y))

Accuracy of the KNN classifier is 95%

As you know the result the accuracy off KNN is highest.