**Mini Project: iris classification**

Done by: Sai Deeraj D (Verzeo ML batch 1)

This project aims to find the accuracy of two data classification algorithms given a dataset of flower species known as iris. Depending on the petals length and width and sepals length and width we basically have 3 types of species (iris setosa , iris versicolor , iris virginica) . We need to classify the species of flower by taking the flowers sepal and petal’s length and width as features and then compare the accuracy of the result obtained.

**Tools used:**

Google Colaboratory - Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Pandas - pandas is a software library written for the Python programming language (or) a python package for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

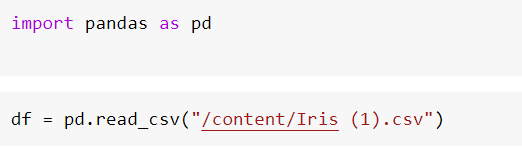
Scikit-learn version 0.24.2 - Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

iris dataset – data of sepal and petal lengths and widths of iris (species of a flower)

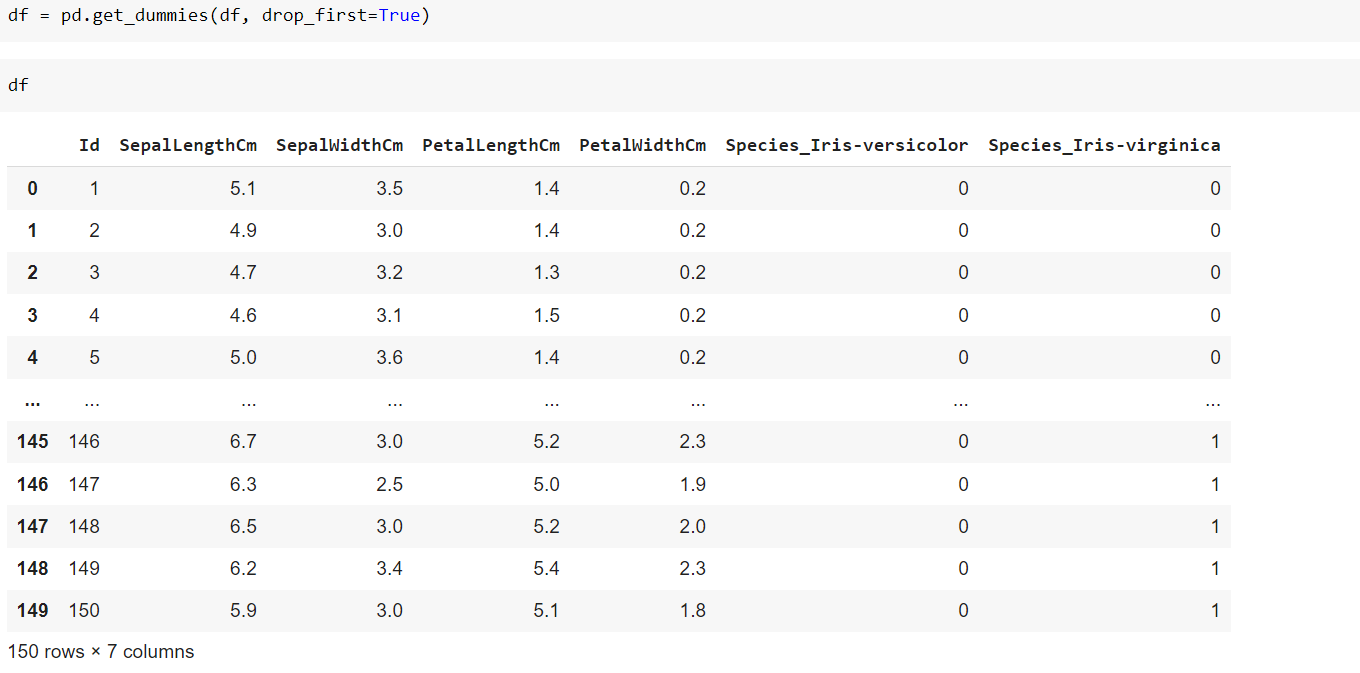
**Algorithm:** (used k- nearest neighbor and decision tree to perform classification)

Step 1: Upload dataset

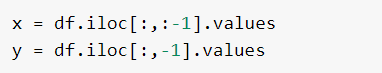
Step 2: Import pandas and read the dataset



Step 3: Convert all the data’s into 0’s and 1’s since our model works on the basis of binary classification otherwise our data won’t be understood by knn and decision tree models imported from our scikit library

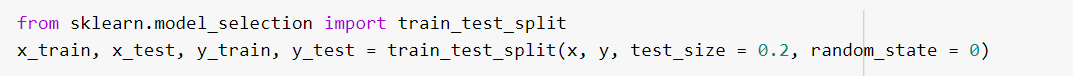


Step 4: Divide our data into inputs and output



Step 5: Split our data for training and testing our model this should be done for both input and output values as well

Note: keep the testing capacity as 20 or 30 % which is standard whereas rest is used for training. While using model ensure that you have imported appropriate model from scikit library



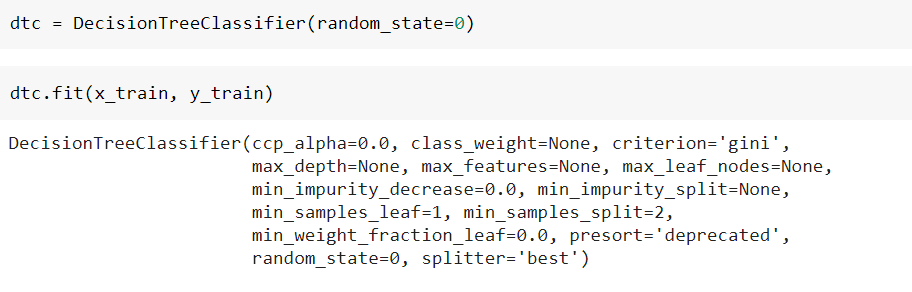
Decision tree implementation:

Step 6: Import decision tree classifier model



Step 7: Create an instance for the class of above model and then fit the data

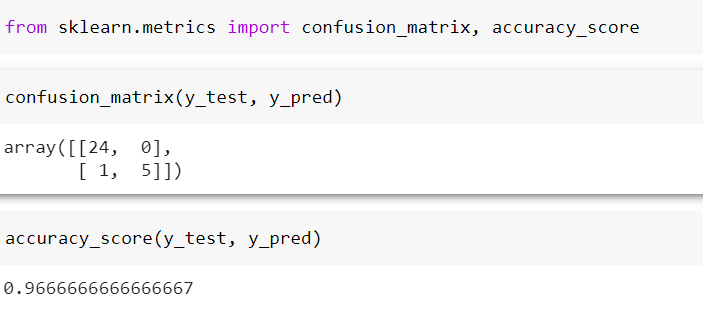
Note: fitting of data must be of training values



Step 8: Now predict the output using the testing values



Step 9: Test the accuracy and get detailed analysis of your models capability

`

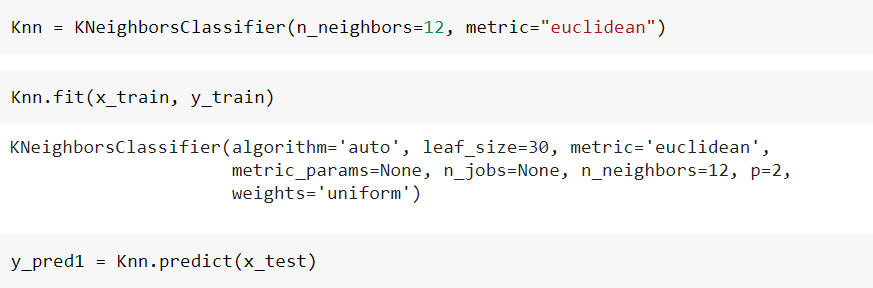
Knn(k – nearest neighbor) implementation:

Step 10: Import knn model



Step 11: Create an instance for the class of above model with appropriate methods you need and then fit the data

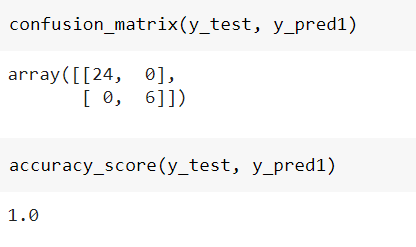
Note: fitting of data must be of training values



Step 8: Now predict the output using the testing values



Step 9: Test the accuracy and get detailed analysis of your models capability



**Conclusion:**

for this dataset knn algorithm is much suitable because the accuracy score is 100 % when compared to decision tree algorithm which has accuracy score as 96.6% which we cannot declare as bad neither.