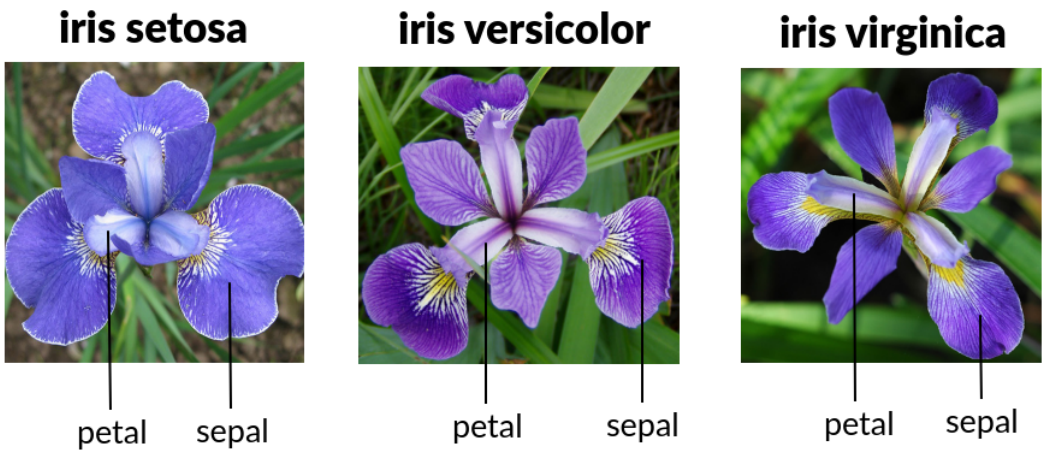
Verzeo Internship - Minor Project - ML June Batch

NAME: Gautam Kumar
SEM: 6th Sem
Dataset: Iris

Project Explanation:

To perform classification analysis on Iris dataset. Perform any two classification algorithms and compare the accuracy. Classification algorithms used are Decision Tree Classifier and KNN Classifier to predict the accuracy of Iris dataset.



Tools Used:

- **Jupyter Notebook:** The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **Sklearn:** Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Algorithms:

- **Decision Tree Classifier:** Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the

given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
DecisionTreeClassifier is a class capable of performing multi-class classification on a dataset.
As with other classifiers, DecisionTreeClassifier takes as input two arrays: an array X, sparse or dense, of shape (n_samples, n_features) holding the training samples, and an array Y of integer values, shape (n_samples,), holding the class labels for the training samples:

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

After being fitted, the model can then be used to predict the class of samples:

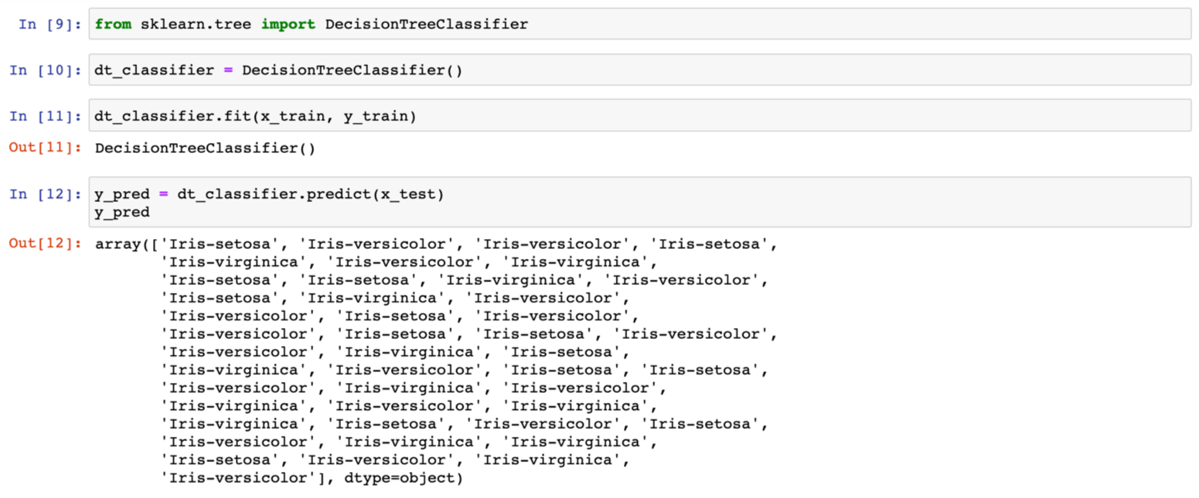
```
>>> clf.predict([[2., 2.]])
array([1])
```

In case that there are multiple classes with the same and highest probability, the classifier will predict the class with the lowest index amongst those classes.
As an alternative to outputting a specific class, the probability of each class can bepredicted, which is the fraction of training samples of the class in a leaf:

```
>>> clf.predict_proba([[2., 2.]])
array([[0., 1.]])
```

DecisionTreeClassifier is capable of both binary (where the labels are [-1, 1]) classification and multiclass (where the labels are [0, …, K-1]) classification.
Using the Iris dataset, we can construct a tree as follows:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> X, y = iris.data, iris.target
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X,y)
```



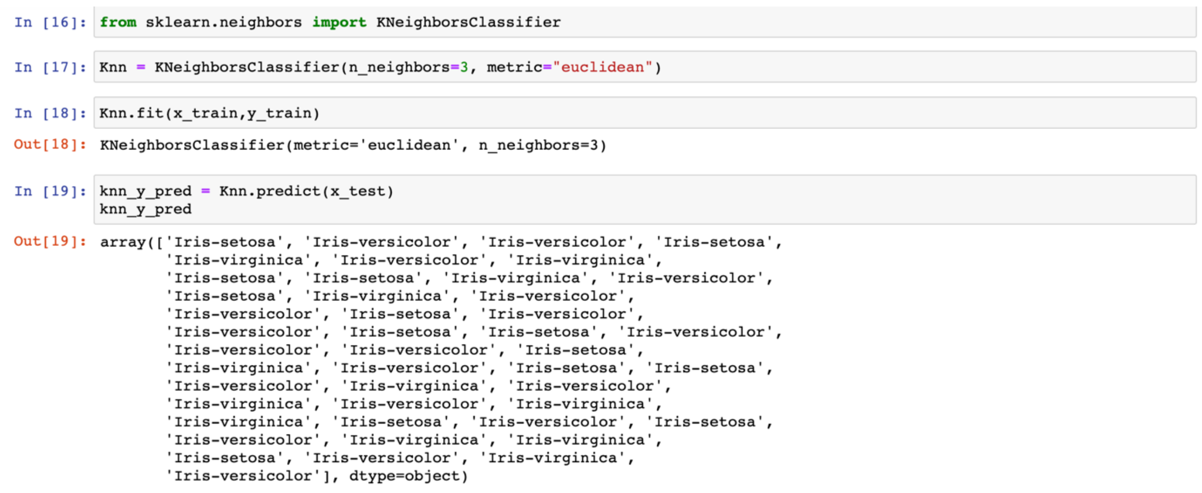
```
In [9]: from sklearn.tree import DecisionTreeClassifier

In [10]: dt_classifier = DecisionTreeClassifier()

In [11]: dt_classifier.fit(x_train, y_train)
Out[11]: DecisionTreeClassifier()

In [12]: y_pred = dt_classifier.predict(x_test)
         y_pred
Out[12]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
                'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
                'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
                'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
                'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
                'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
                'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
                'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
                'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
                'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
                'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
                'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
                'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
                'Iris-versicolor'], dtype=object)
```

- **KNN Classifier:**
  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. K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data. 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. It is extremely straight forward to train the KNN algorithm and make predictions with it, especially when using Scikit-Learn.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

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:

```
y_pred = classifier.predict(X_test)
```



```
In [16]: from sklearn.neighbors import KNeighborsClassifier

In [17]: Knn = KNeighborsClassifier(n_neighbors=3, metric="euclidean")

In [18]: Knn.fit(x_train,y_train)
Out[18]: KNeighborsClassifier(metric='euclidean', n_neighbors=3)

In [19]: knn_y_pred = Knn.predict(x_test)
         knn_y_pred
Out[19]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
                'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
                'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
                'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
                'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
                'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
                'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
                'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
                'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
                'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
                'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
                'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
                'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
                'Iris-versicolor'], dtype=object)
```

Conclusion:

- Accuracy of Iris Dataset using Decision Tree Classifier is 95.55%
- Accuracy of Iris Dataset using KNN Classifier is 97.77%

Accuracy of Iris Dataset varies between Decision Tree Classifier and KNN Classifier in the ratio 0.95:0.97.