Machine Learning (scikit-learn) Project

Artificial Intelligence

MD MAHMUDUL HASAN TAREQ (阿里)

STUDENT ID: 4420190083

Southwest University Of Science And Technology

Computer Science Department

Table of Contents

Abstract

This paper focuses on utilizing machine learning to classify the iris dataset. Actually Machine learning algorithm makes predictions based on previously unknown data or testing data, and it required a set of algorithms to complete the task. There are three types of machine learning are called as Supervised, Unsupervised and Reinforcement learning. In this paper I have worked on supervised learning. I have taken the iris dataset and used DecisionTreeClassification Algorithm. My purpose is build the model that is able to automatically recognize the iris species.

Introduction

Machine learning makes prediction on unseen data or testing data based on train. A computer first learn to perform a task by training dataset in machine learning. Then the computer perform the same task with the testing data. Supervised learning is two types, Classification based and Regression based.

In this paper I am using classification based supervised learning. There are so many algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. I choose the Decision tree classification algorithm because the logic behind the decision tree it's simply understandable and it has a tree-like structure. Decision tree usually mimics human thinking when making a decision.

The implementation of the model includes six basic steps of machine learning that are:

1. Collect data/prepare data
2. Choose algorithm
3. Creating object of the model
4. Train the model by training dataset
5 .Making prediction on unseen data or testing data
6. Evaluation of the model.

I choose the iris dataset because iris dataset is one of the most popular and best known datasets of the machine learning. The iris dataset which can be obtained from UCI Machine Learning Repository and Scikit-Learn . In this paper I took the iris dataset from Scikit-learn (machine learning library) in which iris dataset is already inbuilt.

PRELIMINARY

As I'm going to create a classification model that can reliably recognize iris species automatically. So I prepared data for that, which is data preprocessing and splitting of dataset. Data preprocessing involve handling of missing data, handle of categorical data and handling of feature scaling. Splitting of dataset involves training data and testing data. I shuffle the data so that there is not any particular sequence in training as well as testing dataset. Then I trained my Decision tree classifier algorithm that is a classic supervised machine learning algorithm, which classifies a data by making a tree structure.

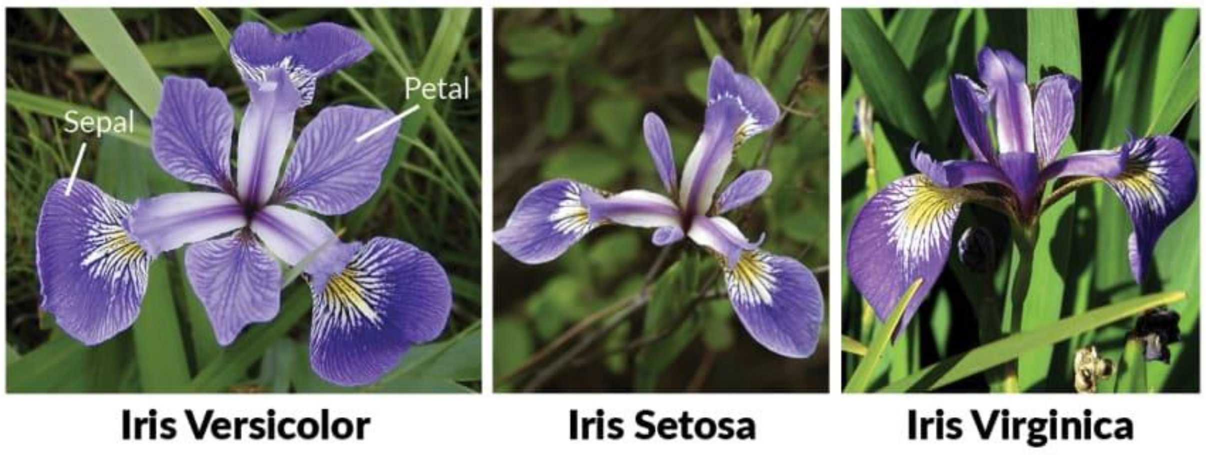
DATASET

The Iris flower data set, also known as Fisher's Iris data set, is a multivariate data set first published in 1936 by British statistician and biologist Ronald Fisher as an example of linear discriminant analysis in his paper on the use of numerous measurements in taxonomic issues.

Dataset Information:
The dataset contain 150 sample data in it. The dataset has three species of data that are Setosa, Versicolor and Virginica each having 50 sample data. Number of attributes in the datasets are four numeric attributes:

1. Sepal length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm

And the fifth attribute is the predictive attributes (class of iris plant) which is include the species of Iris dataset (IRIS Setosa, IRIS Versicolor and IRIS Virginica).



Decision Tree Classification Algorithm

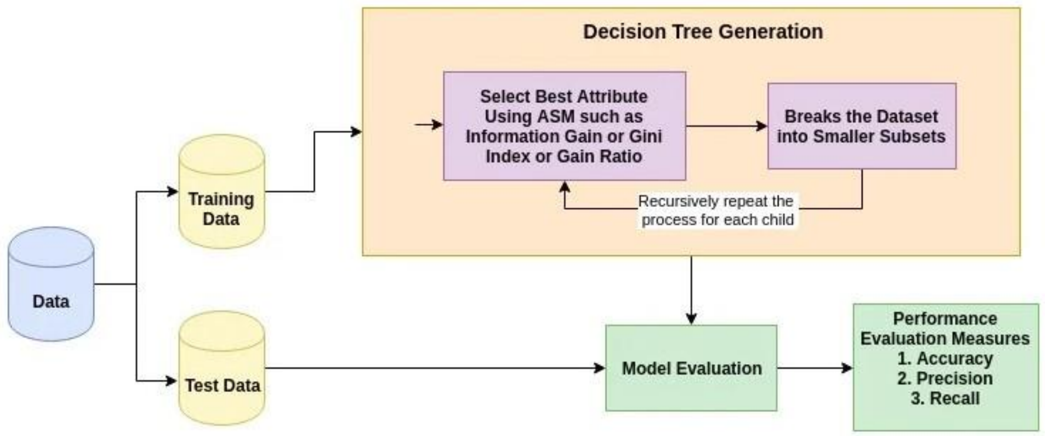
Decision Tree is a supervised learning technique that can be used to solve both classification and regression problems, however it is most commonly used to solve classification problems. Internal nodes represent dataset attributes, branches represent decision rules, and each leaf node provides the outcome in this tree-structured classifier. The Decision Node and the Leaf Node are the two nodes of a Decision tree. 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.

A class that can conduct multi-class classification on a dataset is DecisionTreeClassifier. DecisionTreeClassifier, like other classifiers, takes two arrays as input: a sparse or dense array X of form (n samples, n features) containing the training samples, and an array Y of integer values of shape (n samples, n features) containing the class labels for the training data.The model can be used to predict the class of samples after it has been fitted:

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 be predicted, which is the fraction of training samples of the class in a leaf:

The basic idea behind any decision tree algorithm is as follows:

1. To separate the datasets, first choose the best attribute using Attribute Selection Measures (ASM).
2. Break the dataset into smaller subsets by making that attribute a decision node.
3. Begin tree construction by recursively repeating this approach for each kid until one of the conditions matches:
    - The tuples are all associated with the same attribute value.
    - There are no more qualities available.
    - There are no further instances to be found.


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Implementation and Algorithm Analysis

Here I describe my implementation process and code. I did this code in my google colab research notebook you can find this code by this link:

https://colab.research.google.com/drive/15JNboS-Mx0wanqFgoYWONZp8lX34nl2N?usp=sharing

Step 01: Import all necessary libraries.

Below is the code for importing libraries. I have imported numpy, pandas, seaborn, matplotlib, and some machine learning library from scikit-learn.

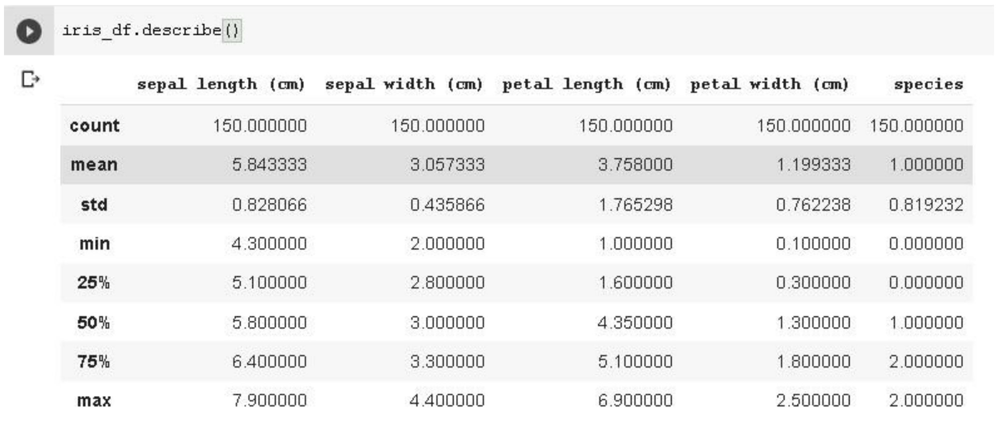
```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import datasets, tree

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

Step 02: Load dataset from scikit-learn.

I have loaded dataset from scikit-learn and view all the information like data, feature, target, attribute information, and then I input my data in a data frame for better work. It can easily represent data in a form naturally suited for data analysis.

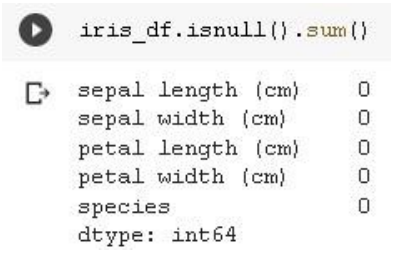
```
iris = datasets.load_iris()

print(iris)

iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)

iris_df.head()

iris_df['species'] = iris.target

iris_df.head(5)

iris_df.describe()

iris_df.isnull().sum()
```

After that I use describe function for better understand my dataset and also I use isnull function for finding null value. Here is the result as follows:



```
iris_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000



```
iris_df.isnull().sum()
```

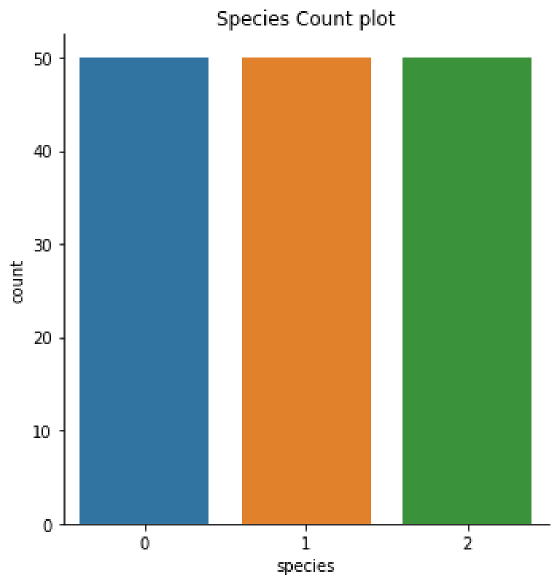
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```

Step 03: Visualize Count plot, Scatter plot and correlation matrix.

Using the 'catplot' function I have plotted the count

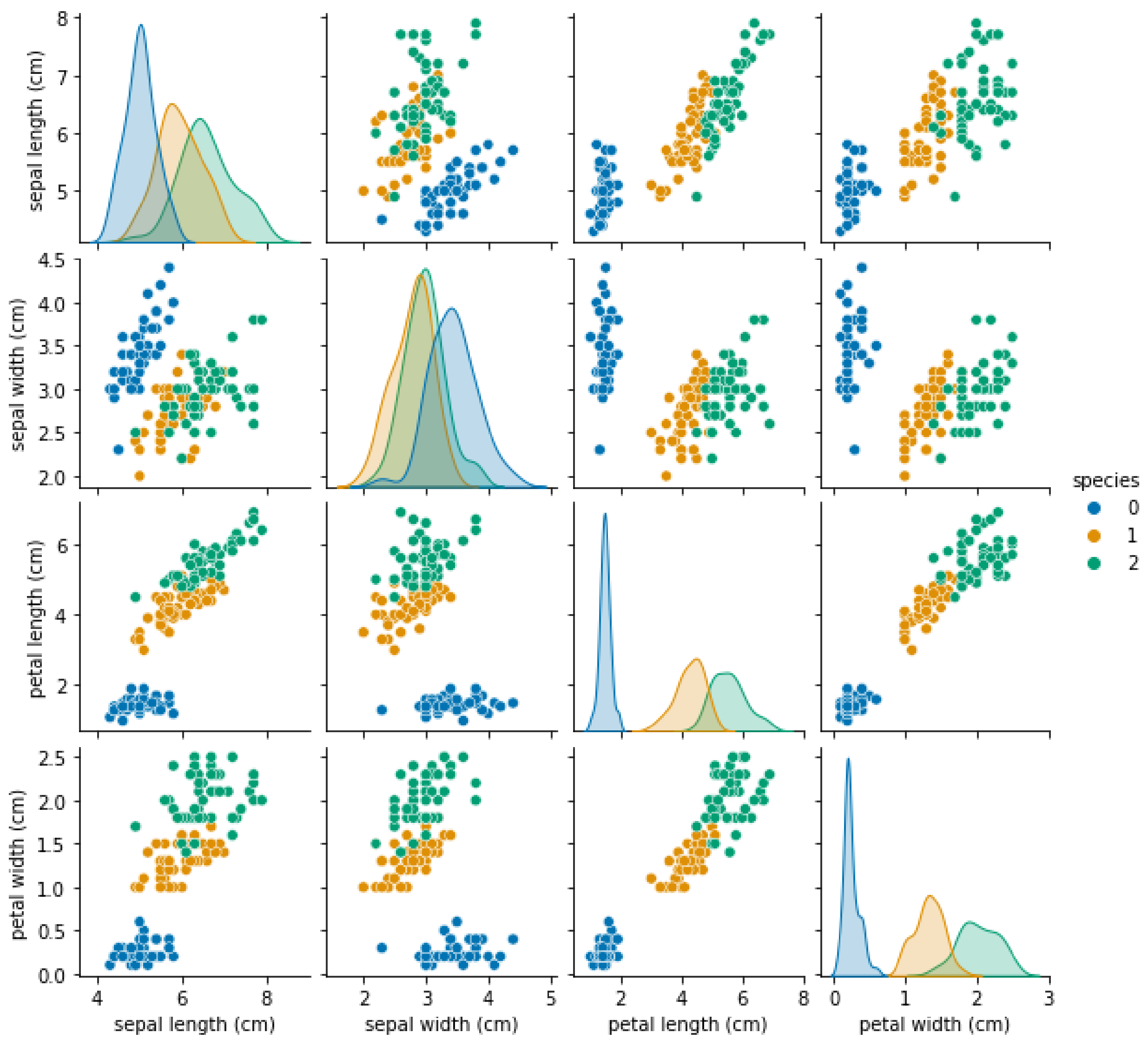
Of each Species which is found to be 50 each.

```
sns.catplot(x='species', data = iris_df, kind = 'count')

plt.title('Species Count plot')
```



I made scatterplots of all-paired attributes by using seaborn’s *pairplot* function.

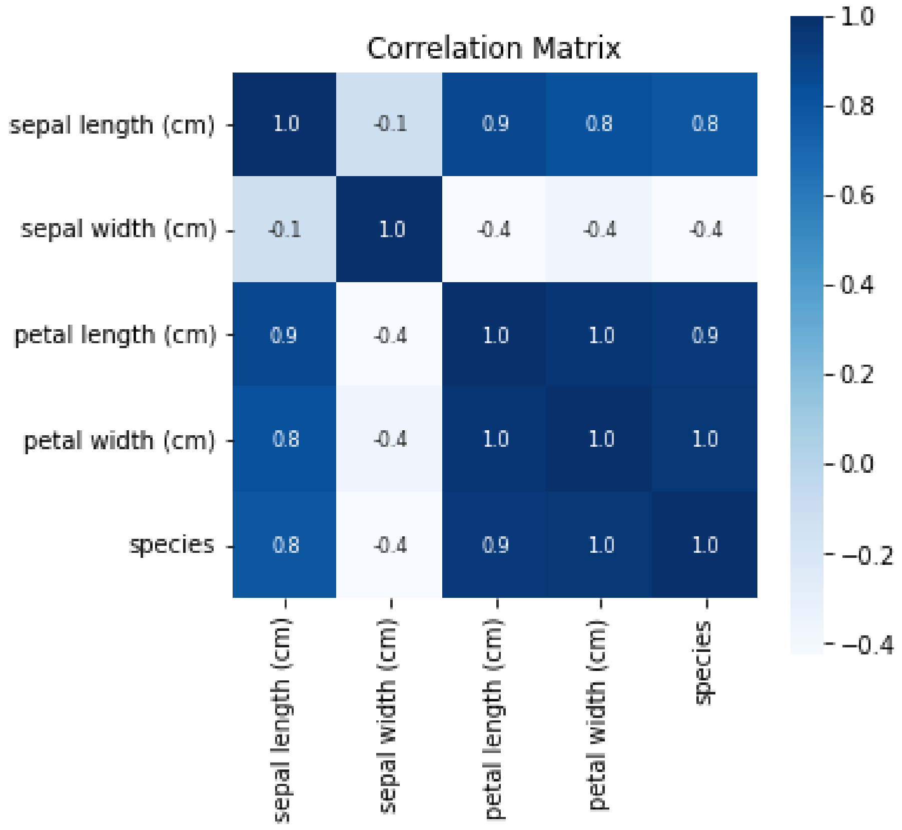
```
sns.pairplot(iris_df, hue="species", height = 2, palette = 'colorblind')
```



Here we can see some variables are highly correlated, e.g., petal_length and petal_width. In addition, the petal measurements separate the different species better than the sepal ones.

Now we can make a correlation matrix to quantitatively examine the relationship between variables:

```
correlation = iris_df.corr()

plt.figure(figsize=(10,10))

plt.title('Correlation Matrix')

sns.heatmap(correlation, cbar=True, square=True, fmt = '.1f', annot = True, annot_kws={'size':8},
```



The most important is that the petal measurements are significantly correlated, whereas the sepal values are uncorrelated. It's worth noting that the petal features have a strong relationship with sepal length, but not with sepal width.

Step 04: Data preprocessing.

In this step first I split my data into labels and features, where X shape is (150, 4) and y shape is (150, 1). Then I standardize my data. And after that I split my data into training dataset and testing dataset, where I took 20% dataset for testing purpose and remaining 80% dataset for training. We know if we can train our model with more data then, our testing result will be more accurate. Below is the code as follows:

```
#Split into labels and features

X = iris_df.drop(columns = 'species', axis=1)

y = iris_df['species']

X.shape,y.shape
```

```
#Standardize our data

standar = StandardScaler()

standar.fit(X)

standardized_data = standar.transform(X)
```

```
#Split into training and testing data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2 ,random_state=1)

X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

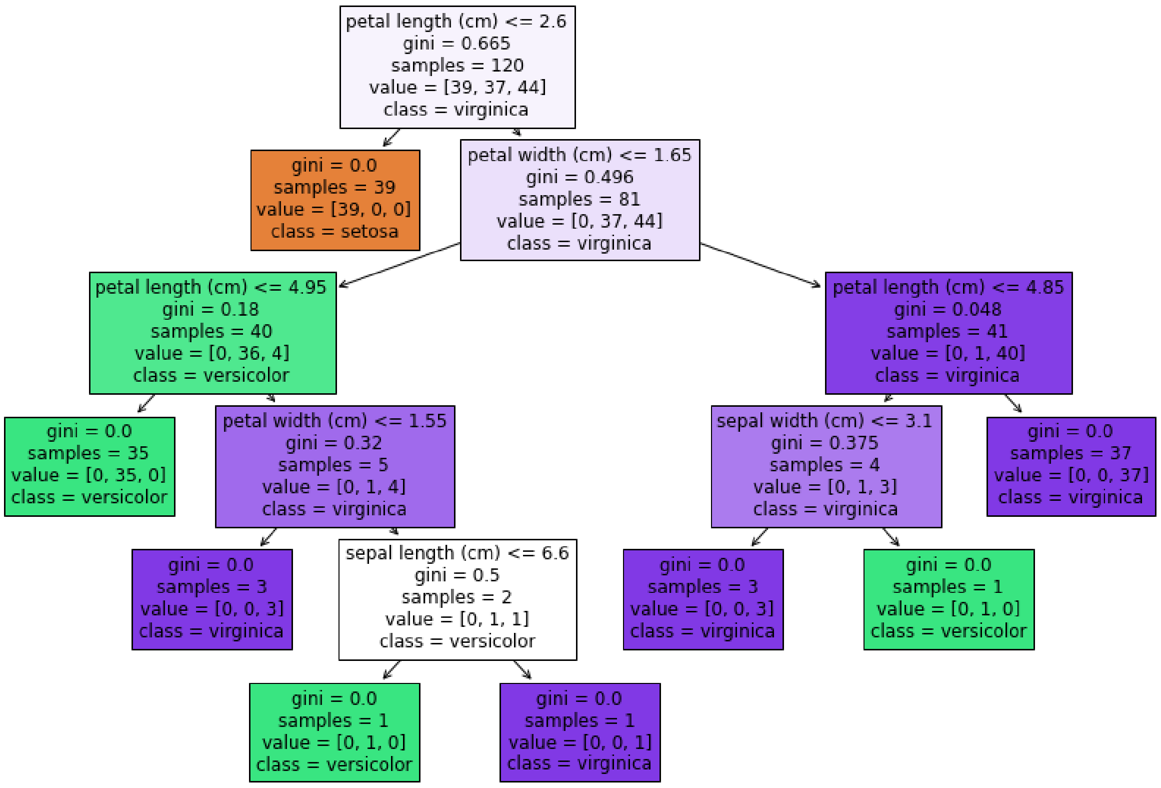
Step 05: Model building and training.

After preprocessing data we can build model. It is considered one of the most important parts of any ML Workflow. I use DecesionTreeClassifier algorithm from scikit-learn and then I trained the algorithm with the training data and training output.

```
iris_model = DecisionTreeClassifier()

iris_model.fit(X_train, y_train)
```

Now I visualized the decision tree classification algorithm so that we can understand how decision tree work.

```
fig = plt.figure(figsize=(15,10))

_ = tree.plot_tree(iris_model,

               feature_names=iris.feature_names,

               class_names=iris.target_names,

               filled=True)
```



Step 06: Predict train accuracy.

By predicting our train accuracy we will know that how correctly our model trained. From train accuracy result we can see that our model trained 100% accurately.

```
Train_predictions = iris_model.predict(X_train)

acctrain = accuracy_score(y_train, Train_predictions)

print("Train accuracy: ", acctrain*100)

Output : Train accuracy:  100.0
```

Step 07: Predict test accuracy.

By predicting our test accuracy we will know that how correctly our model can make a prediction on a dataset based on training. We can see that our test accuracy result is 96.67% which is pretty good testing accuracy. We can say that our model prediction ability is very good. But we can improve more.

```
Test_predictions = iris_model.predict(X_test)

acctest = accuracy_score(y_test, Test_predictions)

print(" Test accuracy: ", acctest*100)

Output : Test accuracy:  96.66666666666667
```

Step 08: Print our all prediction result.

As our dataset is not so big dataset so we can print our all of testing result.

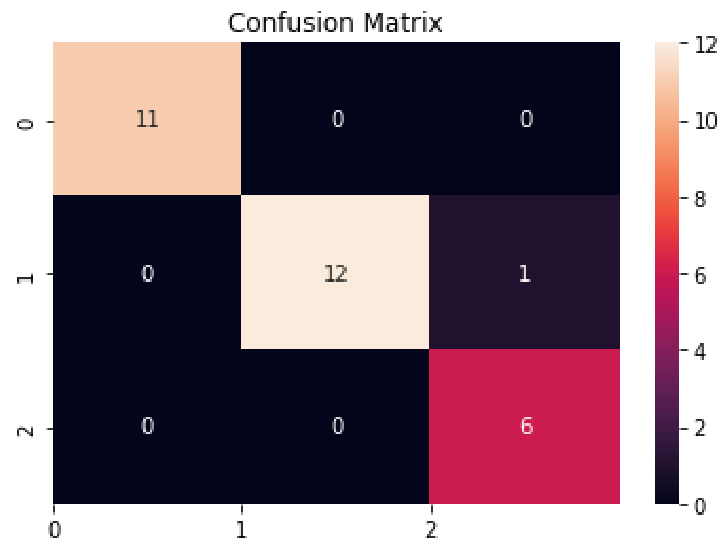
```
classes = ['iris Setosa', 'iris Versicolour', 'iris Virginica']

for i in range(len(Test_predictions)):

   print(classes[Test_predictions[i]])
```

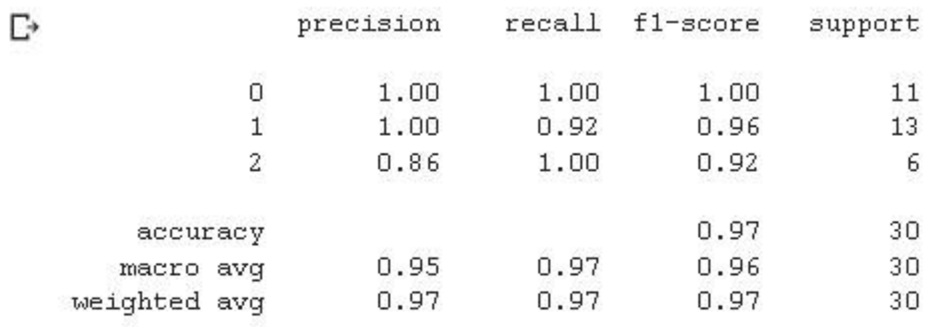
```
iris Setosa
iris Versicolour
iris Versicolour
iris Setosa
iris Virginica
iris Versicolour
iris Virginica
iris Setosa
iris Setosa
iris Virginica
iris Versicolour
iris Setosa
iris Virginica
iris Versicolour
iris Versicolour
iris Setosa
iris Versicolour
iris Versicolour
iris Setosa
iris Setosa
iris Versicolour
iris Versicolour
iris Virginica
iris Setosa
iris Virginica
iris Versicolour
iris Setosa
iris Setosa
iris Versicolour
iris Virginica
```

Step 09: Visualize Confusion Matrix and print classification report.

Here I visualize confusion matrix for understanding our wrong prediction. From this confusion matrix plot we see that there is one versicolor which we predict to be virginica. And also I have print classification report.

```
con_max=confusion_matrix(y_test,Test_predictions)

plt.title('Confusion Matrix')

sns.heatmap(con_max,annot=True)

print(classification_report(y_test,Test_predictions))

plt.xticks(range(3))

plt.show()
```





```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

Step 10: Predict for a single instance.

Finally I made a prediction system which can predict a single instance by taking features data as input. I have taken the iris versicolor features data and paste into input_data option, after run my code I got the correct result.

```
input_data = (6.4,2.9,4.3,1.3)

# changing the input data to a numpy array

input_data_as_numpy_array = np.asarray(input_data)

# reshape the data as we are predicting the label for only one instance

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

predictionTest = iris_model.predict(input_data_reshaped)

print(predictionTest)

if (predictionTest == 0):

  print('Iris setosa')

elif(predictionTest == 1):

  print('Iris versicolor')

else:

    print('Iris verginica')

Output: [1]
Iris versicolor
```

Step 11: End.

Conclusion

I have made a classification model for iris dataset successfully by using machine learning tool from scikit-learn. My classification model is able to recognize the iris species accurately on the basis of 3 species, but some sample provide the wrong prediction. Prediction for setosa and verginica is 100% correct but prediction for versicolor is 3.33% wrong.