Q5) Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set in to training and test set. Compare the accuracy of the different classifiers under the following situations: 5.1 a) Training set=75% Test set=25% b) Training set=66.6%(2/3rd of total),Test set=33.3%

5.2 Training set is chosen by i) hold out method ii) Randomsubsampling iii) Cross-Validation. Compare the accuracy of the classifiers obtained.

5.3 Data is scaled to standard format.

```
import numpy as np  #importing numpy for numpy_array
import pandas as pd  #Importing Pandas for reading CSV files
import matplotlib.pyplot as plt  #importing matplotlib for plotting graphs
import seaborn as sns  #importing seaborn for plotting heatmap
from sklearn.model_selection import train_test_split, cross_val_score  #importing sklearn for
from sklearn.tree import DecisionTreeClassifier, export_text, export_graphviz #importing skle
from sklearn.metrics import accuracy_score, confusion_matrix  #importing sklearn for regressi
from sklearn.neighbors import KNeighborsClassifier  #importing sklearn for regression method
from sklearn.preprocessing import MinMaxScaler  #importing sklearn for regression method
from sklearn.metrics import classification_report
```

```
from google.colab import drive
drive.mount('/content/gdrive')
df = pd.read_csv('/content/gdrive/MyDrive/Q5_iris.csv')
df.info
```

```
     Mounted at /content/gdrive
     <bound method DataFrame.info of      sepal_length  sepal_width  petal_length  petal_widt
     0             5.1          3.5           1.4          0.2     setosa
     1             4.9          3.0           1.4          0.2     setosa
     2             4.7          3.2           1.3          0.2     setosa
     3             4.6          3.1           1.5          0.2     setosa
     4             5.0          3.6           1.4          0.2     setosa
     ..            ...          ...           ...          ...        ...
     145           6.7          3.0           5.2          2.3  virginica
     146           6.3          2.5           5.0          1.9  virginica
     147           6.5          3.0           5.2          2.0  virginica
     148           6.2          3.4           5.4          2.3  virginica
     149           5.9          3.0           5.1          1.8  virginica

     [150 rows x 5 columns]>
```

5.1(a)

```
import sklearn
from sklearn.datasets import *
dir(sklearn.datasets)
```

```
 '_base',
 '_california_housing',
 '_covtype',
 '_kddcup99',
 '_lfw',
 '_olivetti_faces',
 '_openml',
 '_rcv1',
 '_samples_generator',
 '_species_distributions',
 '_svmlight_format_fast',
 '_svmlight_format_io',
 '_twenty_newsgroups',
 'clear_data_home',
 'dump_svmlight_file',
 'fetch_20newsgroups',
 'fetch_20newsgroups_vectorized',
 'fetch_california_housing',
 'fetch_covtype',
 'fetch_kddcup99',
 'fetch_lfw_pairs',
 'fetch_lfw_people',
 'fetch_olivetti_faces',
 'fetch_openml',
 'fetch_rcv1',
 'fetch_species_distributions',
 'get_data_home',
 'load_boston',
 'load_breast_cancer',
 'load_diabetes',
 'load_digits',
 'load_files',
 'load_iris',
 'load_linnerud',
 'load_sample_image',
 'load_sample_images',
 'load_svmlight_file',
 'load_svmlight_files',
 'load_wine',
 'make_biclusters',
 'make_blobs',
 'make_checkerboard',
 'make_circles',
 'make_classification',
 'make_friedman1',
 'make_friedman2',
 'make_friedman3',
 'make_gaussian_quantiles',
 'make_hastie_10_2',
 'make_low_rank_matrix',
 'make_moons',
 'make_multilabel_classification',
 'make_regression',
 'make_s_curve',
 'make_sparse_coded_signal',
 'make_sparse_spd_matrix',
 'make_sparse_uncorrelated',
  'make_sparse_uncorrelated',
 'make_spd_matrix',
 'make_swiss_roll']
```

5.1(b)

```
df=load_iris()
df.keys()
```

```
    dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

```
print(df.filename)
print(df.feature_names)
```

```
    /usr/local/lib/python3.7/dist-packages/sklearn/datasets/data/iris.csv
    ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
df = pd.read_csv('/content/gdrive/MyDrive/Q5_iris.csv')
print(df)
```

```
         sepal_length  sepal_width  petal_length  petal_width    species
    0             5.1          3.5           1.4          0.2     setosa
    1             4.9          3.0           1.4          0.2     setosa
    2             4.7          3.2           1.3          0.2     setosa
    3             4.6          3.1           1.5          0.2     setosa
    4             5.0          3.6           1.4          0.2     setosa
    ..            ...          ...           ...          ...        ...
    145           6.7          3.0           5.2          2.3  virginica
    146           6.3          2.5           5.0          1.9  virginica
    147           6.5          3.0           5.2          2.0  virginica
    148           6.2          3.4           5.4          2.3  virginica
    149           5.9          3.0           5.1          1.8  virginica

    [150 rows x 5 columns]
```

```
df=df.drop_duplicates()
```

```
X=df.values[:,:-1]
Y=df.values[:,-1]
```

```
print(X.shape)
print(Y.shape)
```

```
    (147, 4)
    (147,)
```

```
import numpy as np
classLabels=np.unique(Y)
classLabels
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
 test_val = 0.25
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=test_val)
```

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

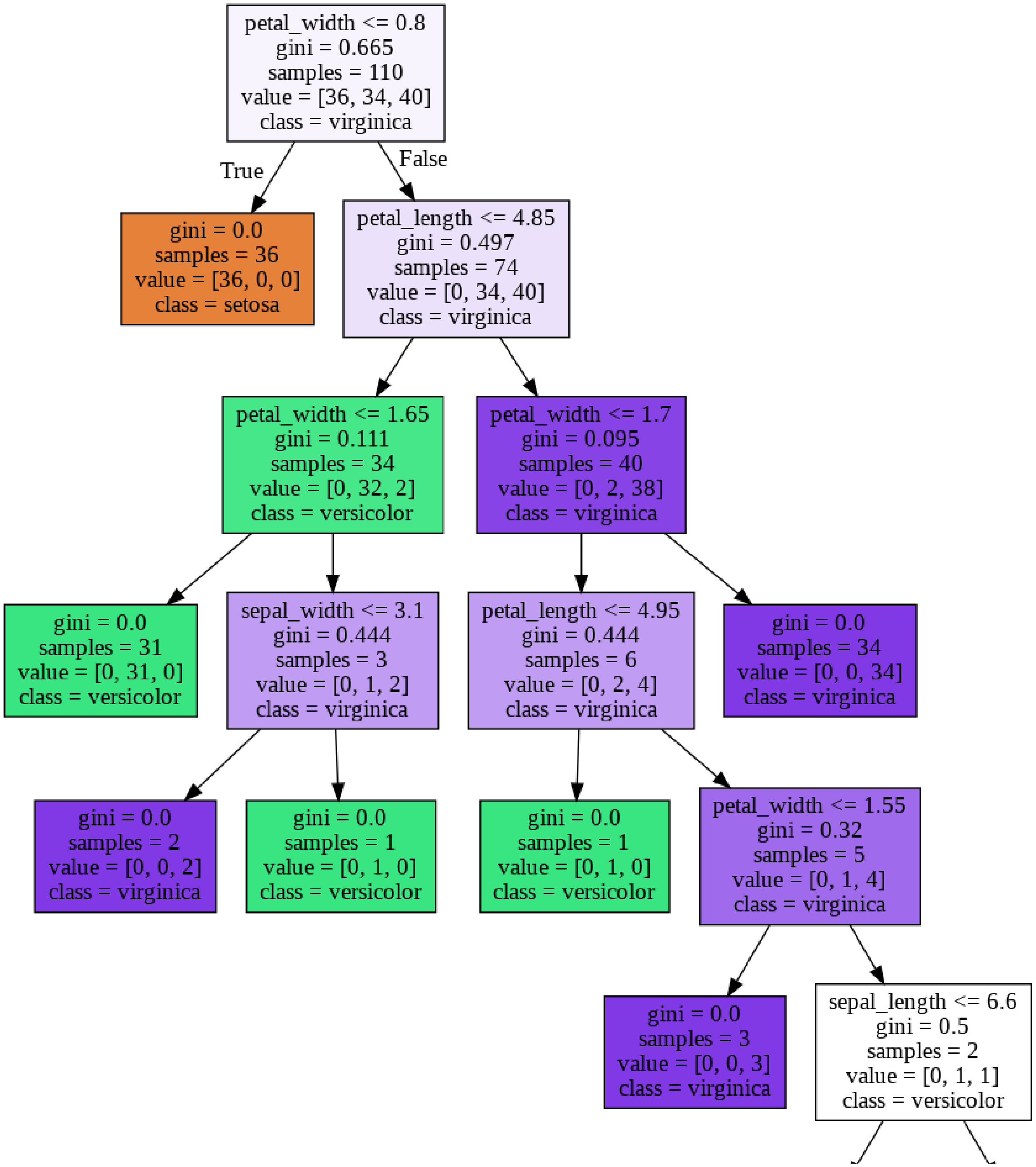
```
(110, 4)
(37, 4)
(110,)
(37,)
```

Decision Tree Classifier

5.2

```
DTclassifer = DecisionTreeClassifier().fit(X_train,Y_train)
```

```
 import pydotplus
from IPython.display import Image
dot_data=export_graphviz(DTclassifer, feature_names=df.columns[:-1], class_names=classLabels,
graph=pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



```
 print(Y_test)
Y_predict=DTclassifer.predict(X_test)
print(Y_predict)
```

```
    ['setosa' 'versicolor' 'setosa' 'versicolor' 'virginica' 'setosa'
     'virginica' 'versicolor' 'virginica' 'virginica' 'setosa' 'virginica'
     'setosa' 'virginica' 'virginica' 'virginica' 'versicolor' 'versicolor'
     'versicolor' 'versicolor' 'versicolor' 'versicolor' 'setosa' 'versicolor'
     'setosa' 'setosa' 'versicolor' 'setosa' 'setosa' 'versicolor'
     'versicolor' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'setosa'
     'virginica']
    ['setosa' 'virginica' 'setosa' 'versicolor' 'virginica' 'setosa'
     'virginica' 'versicolor' 'virginica' 'virginica' 'setosa' 'virginica'
     'setosa' 'virginica' 'virginica' 'virginica' 'versicolor' 'versicolor'
     'versicolor' 'versicolor' 'versicolor' 'versicolor' 'setosa' 'versicolor'
```

```
 'setosa' 'setosa' 'versicolor' 'setosa' 'setosa' 'versicolor'
 'versicolor' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'setosa'
 'virginica']
```

```
accuracy_score(Y_test, Y_predict)
```

```
0.972972972972973
```

```
confusematr=confusion_matrix(Y_test, Y_predict)
confusematr
```

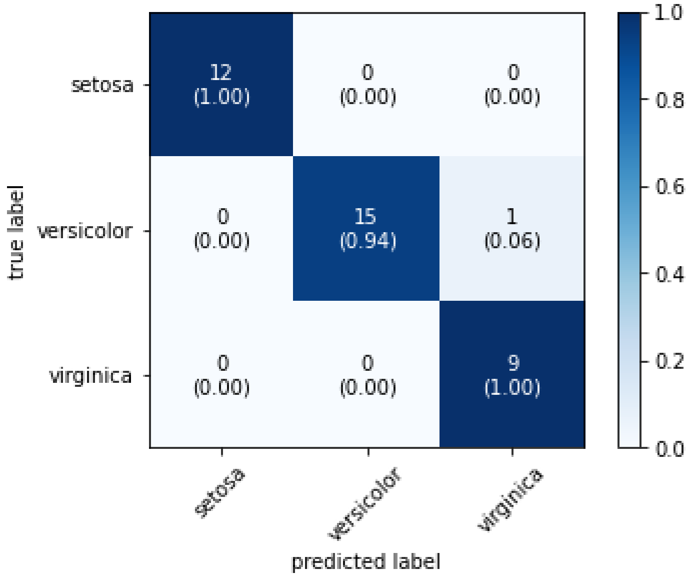
```
array([[12,  0,  0],
       [ 0, 15,  1],
       [ 0,  0,  9]])
```

```
!pip install mlxtend --upgrade --no-deps
```

```
Requirement already up-to-date: mlxtend in /usr/local/lib/python3.7/dist-packages (0.18
```

```
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(conf_mat=confusematr, colorbar=True, show_absolute=True, show_normed=Tr
```

```
(<Figure size 432x288 with 2 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f1a8dda3d90>)
```

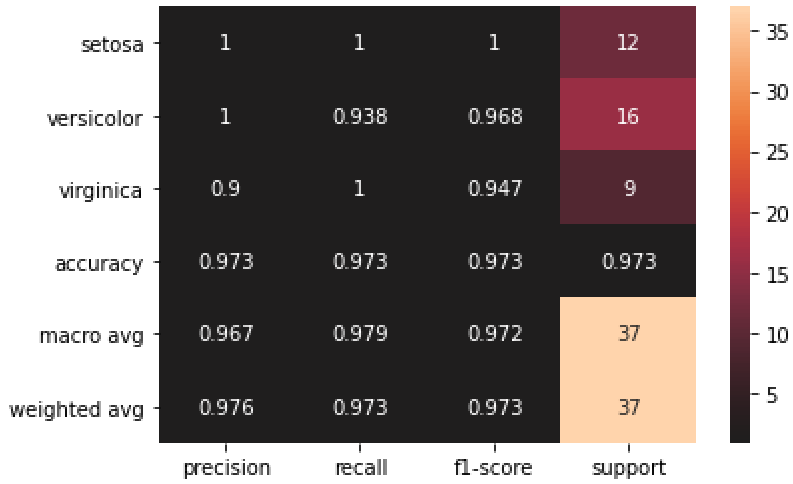


```
clf_report=classification_report(Y_test, Y_predict, target_names=classLabels, digits=5, outpu
print(clf_report)
```

```
{'setosa': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 12}, 'versicol
```

```
import seaborn as sns
```

```
sns.heatmap(pd.DataFrame(clf_report).T, annot=True, fmt='.3g',center=True)

     <matplotlib.axes._subplots.AxesSubplot at 0x7f1a8dd3b950>
```



```
numTimes=10

accuracy=list()
for i in range(numTimes):
  X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=test_val)
  DTclassifer = DecisionTreeClassifier(criterion="entropy").fit(X_train,Y_train)
  Y_predict=DTclassifer.predict(X_test)
  accuracy.append(accuracy_score(Y_test, Y_predict))
print(sum(accuracy)/numTimes)

     0.9540540540540541
```

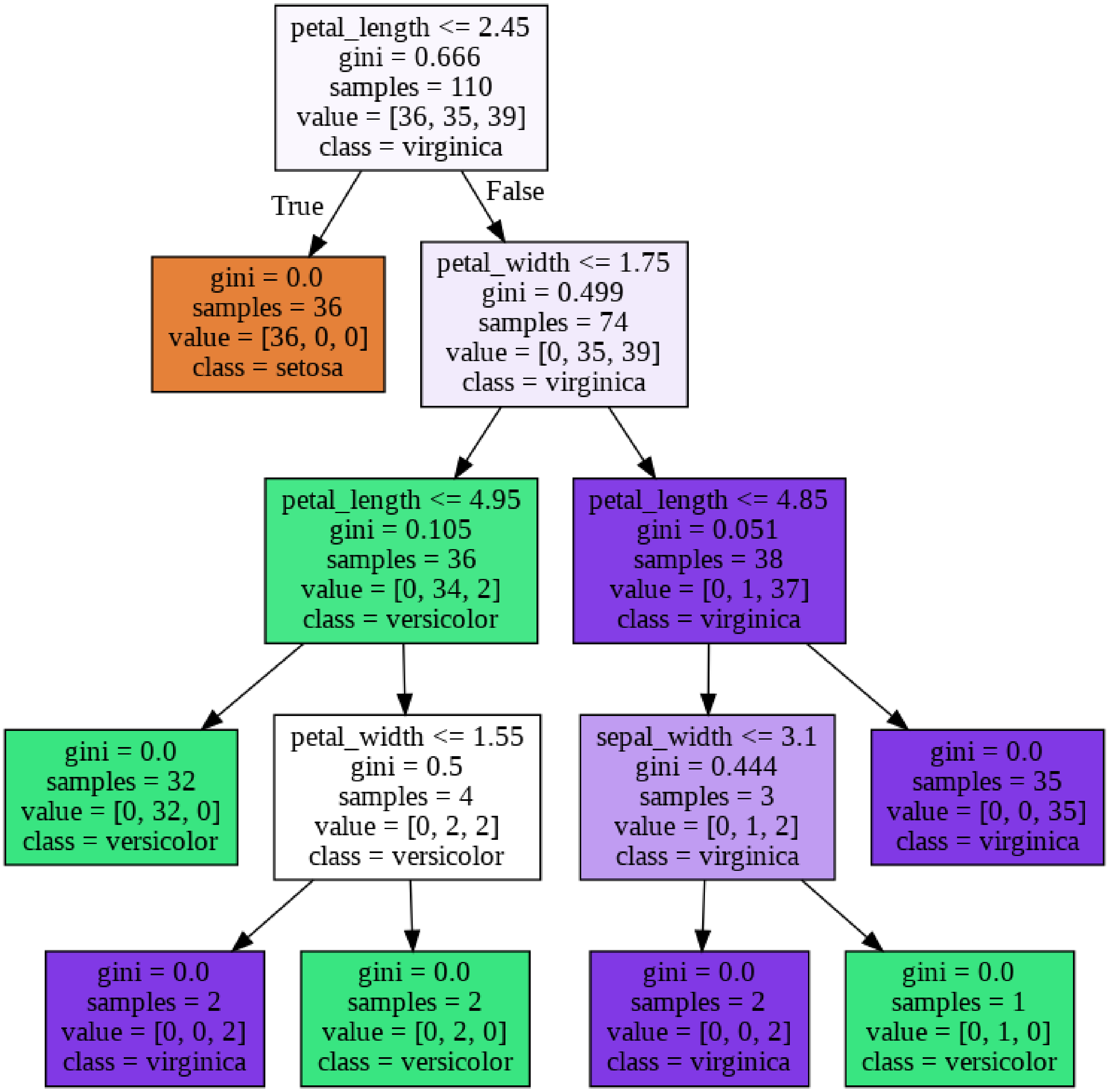
```
k=10
scores=cross_val_score(DecisionTreeClassifier(),X,Y,cv=k)
print(scores,scores.mean())

     [1.         0.93333333 1.         0.93333333 0.93333333 0.86666667
      0.93333333 1.         1.         1.        ] 0.96
```

```
test_val1 =(1/3)
X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X,Y,test_size=test_val)


print(X_train1.shape)
print(X_test1.shape)
print(Y_train1.shape)
print(Y_test1.shape)

     (110, 4)
     (37, 4)
     (110,)
     (37,)
```

New Data Classifier

```
DTclassifer1 = DecisionTreeClassifier().fit(X_train1,Y_train1)
```

```
import pydotplus
from IPython.display import Image
dot_data1=export_graphviz(DTclassifer1, feature_names=df.columns[:-1], class_names=classLabel
graph1=pydotplus.graph_from_dot_data(dot_data1)
Image(graph1.create_png())
```



```
print(Y_test1)
Y_predict1=DTclassifer1.predict(X_test1)
print(Y_predict1)
```

```
['setosa' 'setosa' 'versicolor' 'versicolor' 'virginica' 'virginica'
 'virginica' 'versicolor' 'virginica' 'setosa' 'setosa' 'setosa'
 'virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'virginica' 'setosa' 'versicolor' 'versicolor' 'setosa' 'versicolor'
 'virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'setosa'
 'versicolor' 'virginica' 'versicolor' 'setosa' 'virginica' 'setosa'
 'versicolor']
['setosa' 'setosa' 'versicolor' 'versicolor' 'virginica' 'virginica'
 'virginica' 'versicolor' 'versicolor' 'setosa' 'setosa' 'setosa'
 'virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'virginica' 'setosa' 'versicolor' 'versicolor' 'setosa' 'versicolor'
 'virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'setosa'
 'versicolor' 'virginica' 'versicolor' 'setosa' 'versicolor' 'setosa'
 'versicolor']
```

```
accuracy_score(Y_test1, Y_predict1)
```

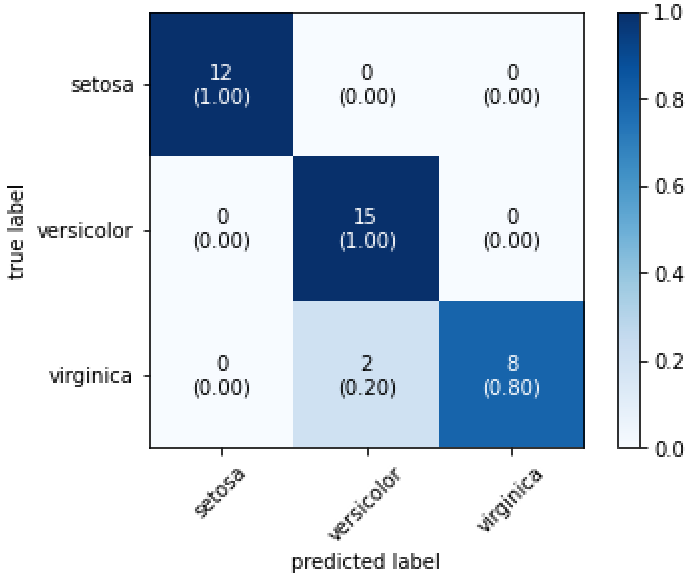
```
0.9459459459459459
```

```
confusematr1=confusion_matrix(Y_test1, Y_predict1)
confusematr1
```

```
array([[12,  0,  0],
       [ 0, 15,  0],
       [ 0,  2,  8]])
```

```
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(conf_mat=confusematr1, colorbar=True, show_absolute=True, show_normed=T
```

```
(<Figure size 432x288 with 2 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f1a83484cd0>)
```

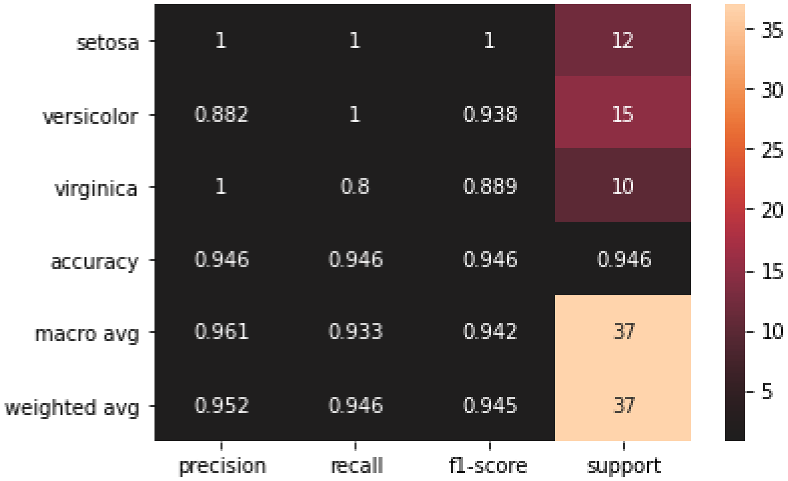


```
clf_report1=classification_report(Y_test1, Y_predict1, target_names=classLabels, digits=5, ou
print(clf_report1)
```

```
{'setosa': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 12}, 'versicolo
```

```
import seaborn as sns
sns.heatmap(pd.DataFrame(clf_report1).T, annot=True, fmt='.3g',center=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1a8338f0d0>
```



```
numTimes=10

accuracy=list()
for i in range(numTimes):
  X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X,Y,test_size=test_val)
  DTclassifer1 = DecisionTreeClassifier(criterion="entropy").fit(X_train1,Y_train1)
  Y_predict1=DTclassifer1.predict(X_test1)
  accuracy.append(accuracy_score(Y_test1, Y_predict1))
print(sum(accuracy)/numTimes)
```

```
0.9432432432432434
```

```
k=10
scores1=cross_val_score(DecisionTreeClassifier(),X,Y,cv=k)
print(scores1,scores.mean())
```

```
[1.         0.93333333 1.         0.93333333 0.93333333 0.86666667
 0.93333333 0.92857143 1.         1.        ] 0.96
```