In [1]:
```
import pandas as pd
df = pd.read_csv('https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv')
df
```

Out[1]:

	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 × 5 columns

In [2]:
```
#Removing duplicates from the data frame
df = df.drop_duplicates()
```

In [3]:
```
#Seperating the target variable(value to be predicted)
Y = df['species']  #Variable to be predicted
X = df[['petal_length','petal_width','sepal_length','sepal_width']]
```

In [4]:
```
from sklearn import preprocessing
columns_tb_scaled = []
for i in X.columns:
  if X[i].mean()!=0 and X[i].std()!=1:
    columns_tb_scaled.append(i)

X = pd.DataFrame(preprocessing.scale(X[columns_tb_scaled]),columns=columns_tb_scaled) #Scaling the appropriate data
X
```

Out[4]:

	petal_length	petal_width	sepal_length	sepal_width
0	-1.357737	-1.335700	-0.915509	1.019971
1	-1.357737	-1.335700	-1.157560	-0.128082
2	-1.414778	-1.335700	-1.399610	0.331139
3	-1.300696	-1.335700	-1.520635	0.101529
4	-1.357737	-1.335700	-1.036535	1.249582
...	...	...	...	...
142	0.809831	1.444682	1.020892	-0.128082
143	0.695748	0.915085	0.536792	-1.276136
144	0.809831	1.047484	0.778842	-0.128082
145	0.923913	1.444682	0.415766	0.790361
146	0.752789	0.782686	0.052691	-0.128082

147 rows × 4 columns

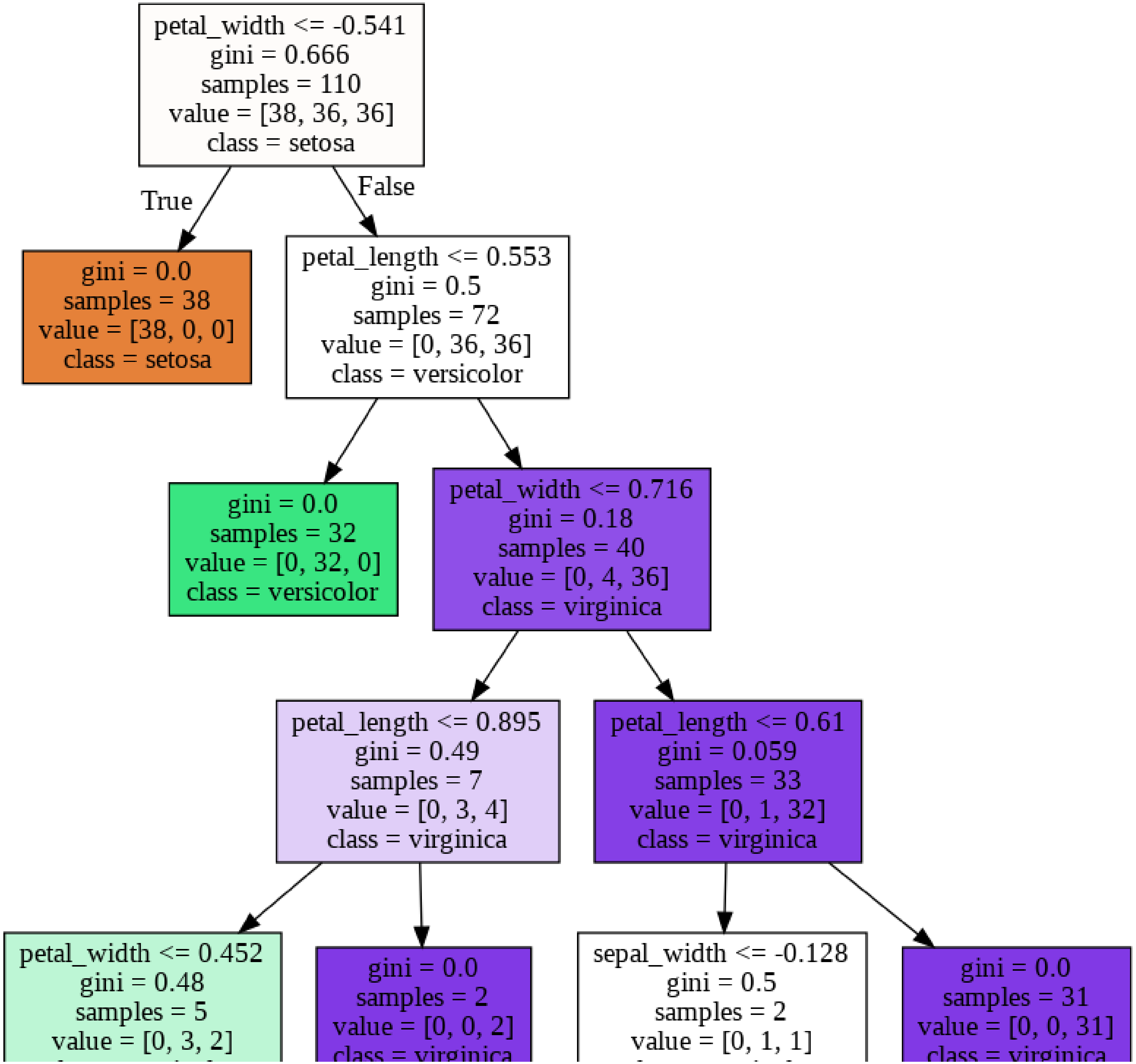
In [5]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25) #Splitting the data for testing(25%) and training
(75%)
```

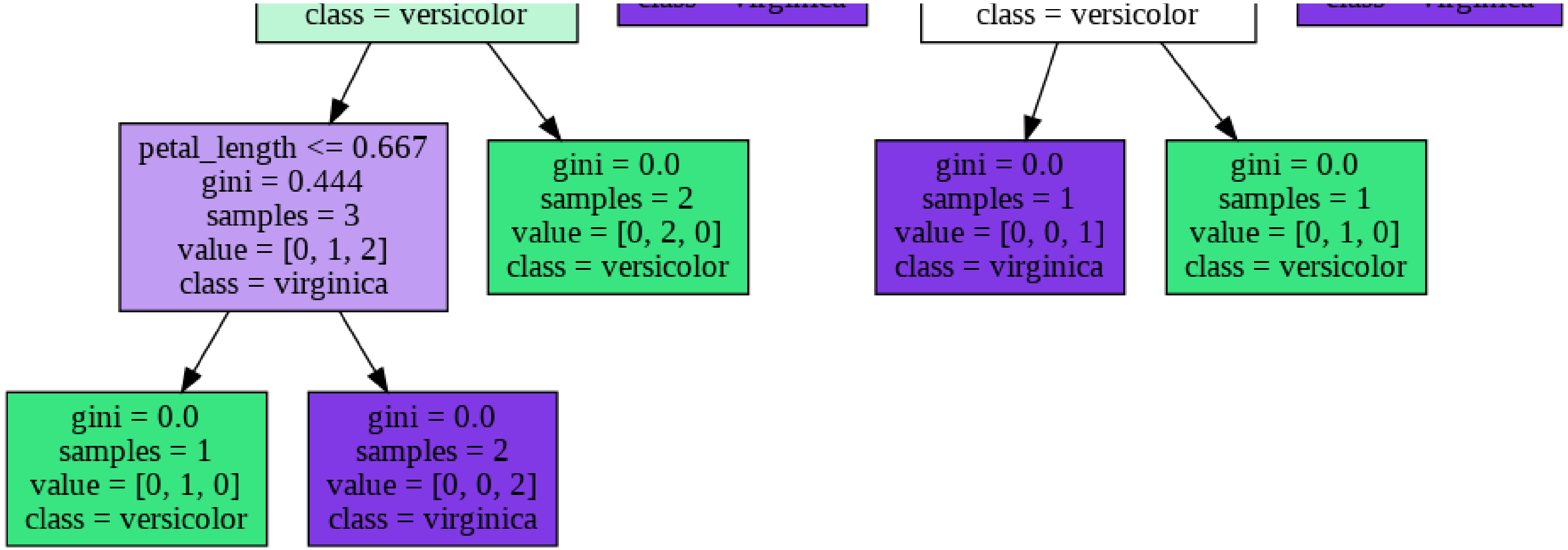
In [6]:
```
#Training the model on our training data
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree.fit(X_train,Y_train)
```

Out[6]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [7]:
```
#Displaying decision tree
import pydotplus
from IPython.display import Image
from sklearn.tree import export_graphviz,export_text

graph = export_graphviz(tree,feature_names=X.columns,class_names=Y.unique(),filled=True)
graph = pydotplus.graph_from_dot_data(graph)
Image(graph.create_png())
```

Out[7]:


petal_width <= -0.541
gini = 0.666
samples = 110
value = [38, 36, 36]
class = setosa
True
False
gini = 0.0
samples = 38
value = [38, 0, 0]
class = setosa
petal_length <= 0.553
gini = 0.5
samples = 72
value = [0, 36, 36]
class = versicolor
gini = 0.0
samples = 32
value = [0, 32, 0]
class = versicolor
petal_width <= 0.716
gini = 0.18
samples = 40
value = [0, 4, 36]
class = virginica
petal_length <= 0.895
gini = 0.49
samples = 7
value = [0, 3, 4]
class = virginica
petal_length <= 0.61
gini = 0.059
samples = 33
value = [0, 1, 32]
class = virginica
petal_width <= 0.452
gini = 0.48
samples = 5
value = [0, 3, 2]
gini = 0.0
samples = 2
value = [0, 0, 2]
sepal_width <= -0.128
gini = 0.5
samples = 2
value = [0, 1, 1]
gini = 0.0
samples = 31
value = [0, 0, 31]



In [8]:

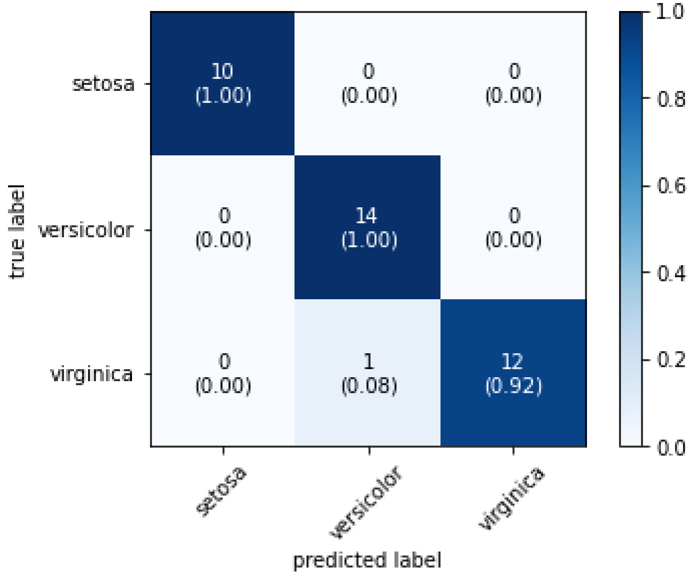
```
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
Y_pred = tree.predict(X_test)
print("Accuracy of the tree:",accuracy_score(Y_test,Y_pred)*100)

print("Confusion matrix:")
mat = confusion_matrix(Y_test,Y_pred)
print(mat)
report=classification_report(Y_test,Y_pred,target_names=Y.unique(), output_dict=True)
print("Classification report",report)
```

```
Accuracy of the tree: 97.2972972972973
Confusion matrix:
[[10  0  0]
 [ 0 14  0]
 [ 0  1 12]]
Classification report {'setosa': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 10}, 'versicolor': {'p
recision': 0.9333333333333333, 'recall': 1.0, 'f1-score': 0.9655172413793104, 'support': 14}, 'virginica': {'precisio
n': 1.0, 'recall': 0.9230769230769231, 'f1-score': 0.9600000000000001, 'support': 13}, 'accuracy': 0.972972972972973,
'macro avg': {'precision': 0.9777777777777779, 'recall': 0.9743589743589745, 'f1-score': 0.9751724137931035, 'suppor
t': 37}, 'weighted avg': {'precision': 0.9747747747747747, 'recall': 0.972972972972973, 'f1-score': 0.972898415657036
4, 'support': 37}}
```

In [9]:
```
!pip install mlxtend --upgrade --no-deps
from mlxtend.plotting import plot_confusion_matrix
import numpy as np
plot_confusion_matrix(conf_mat=mat, colorbar=True, show_absolute=True, show_normed=True, class_names=Y.unique())
```

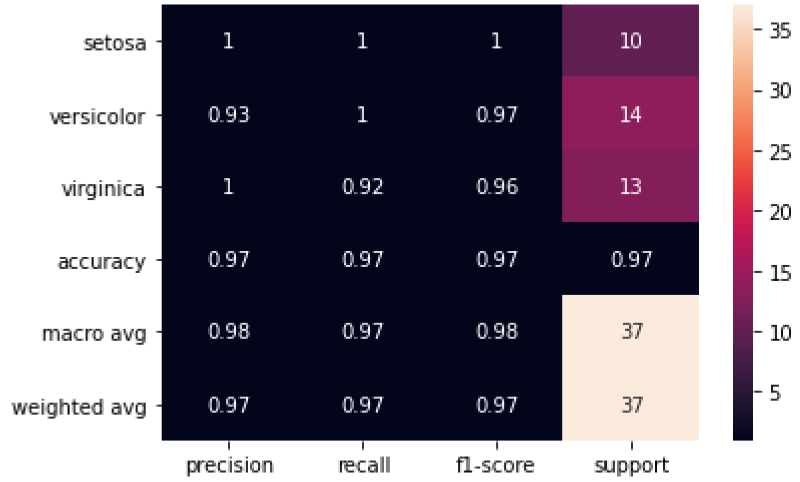
```
Requirement already up-to-date: mlxtend in /usr/local/lib/python3.7/dist-packages (0.18.0)
```

Out[9]:
```
(<Figure size 432x288 with 2 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f83aee53210>)
```



In [10]:
```
import seaborn as sns
sns.heatmap(pd.DataFrame(report).T,annot=True)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f83aee656d0>



In [11]:
```
#Random sampling
limit=15
Ratio=0.25
acc=[]
for i in range(limit):
  X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=Ratio)
  clf=DecisionTreeClassifier(criterion="entropy")
  clf=clf.fit(X_train,Y_train)
  Y_pred=clf.predict(X_test)
  acc.append(accuracy_score(Y_test,Y_pred))
print(sum(acc)/limit*100)
```

```
93.6936936936937
```

In [12]:
```
#K-cross validation
from sklearn.model_selection import cross_val_score
k=10
score = cross_val_score(DecisionTreeClassifier(),X,Y,cv=k)

print("Accuracy of 10 splits:",score*100)
print("Mean accuracy:",score.mean()*100)
```

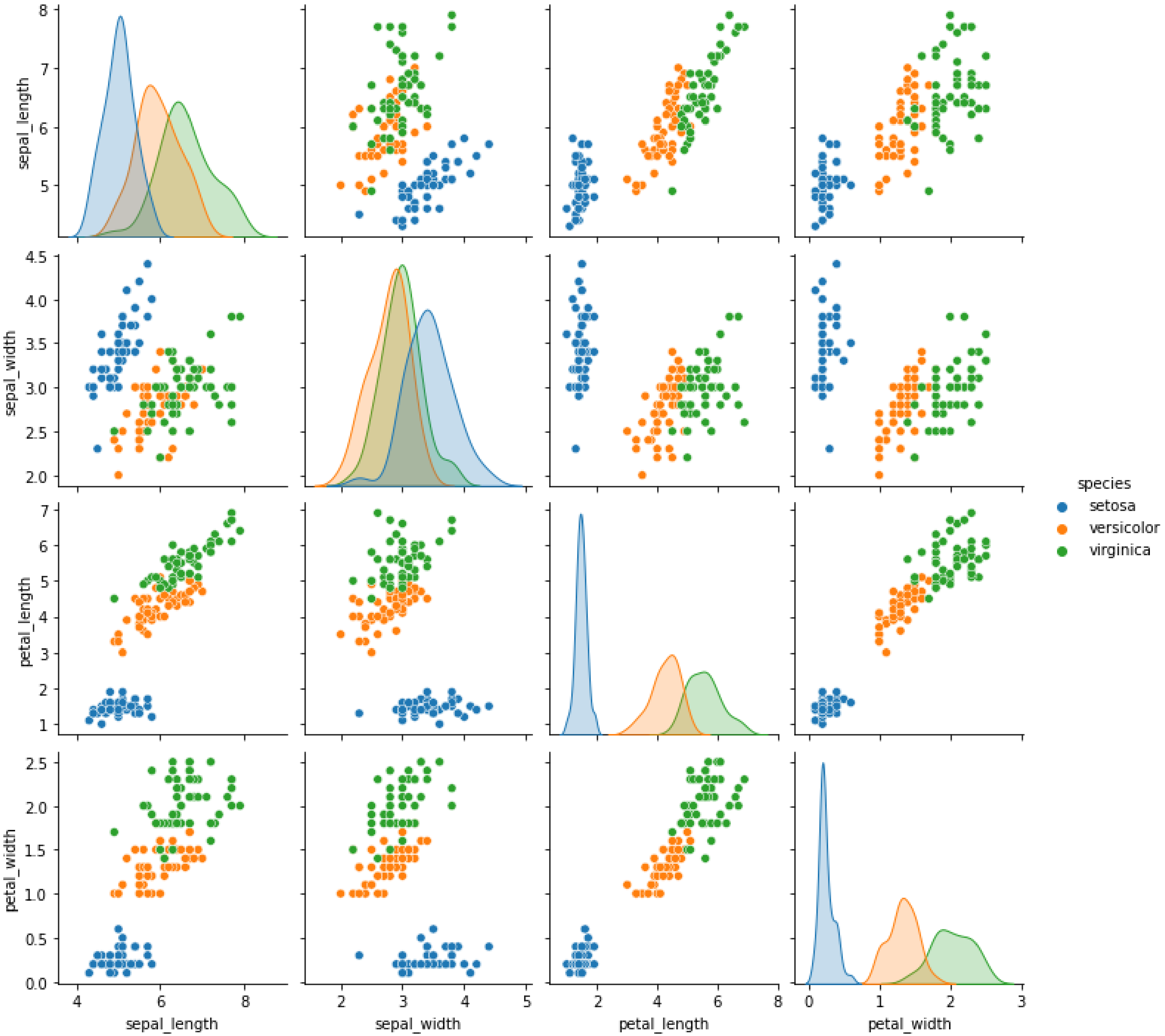
```
Accuracy of 10 splits: [100.          93.33333333 100.          93.33333333  93.33333333
  86.66666667  93.33333333  92.85714286 100.         100.        ]
Mean accuracy: 95.28571428571428
```

In [13]:
```
#For naive bayes classifer
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
```

In [14]:
```
#For K-nearest neighours
from sklearn.neighbors import KNeighborsClassifier
kn = KNeighborsClassifier(n_neighbors = 3)
```

In [15]:
```
#Scatterplot for iris dataset
sns.pairplot(df,hue='species')
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x7f83ad27b210>


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

In [16]:
```
from sklearn.metrics import classification_report

def Accu_conf(clf,X,Y):
  #For holdout method
  X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
  clf.fit(X_train,Y_train)

  Y_pred = clf.predict(X_test)
  report=classification_report(Y_test,Y_pred,target_names=Y.unique(), output_dict=True)

  print("Accuracy:",accuracy_score(Y_test,Y_pred)*100)
  print("Confusion matrix:")
  print(confusion_matrix(Y_test,Y_pred))
  print("Classification report:\n",report)
```

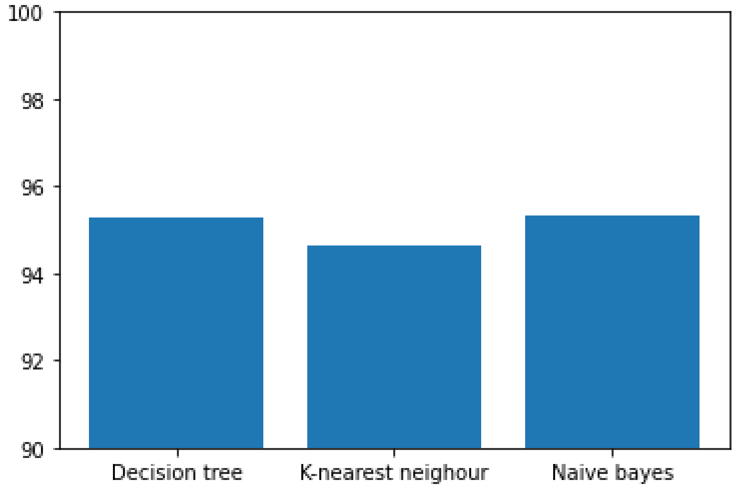
In [17]:
```
#Accuracy and confusion matrix for  different classifiers
print("For K-nearest neighours:")
Accu_conf(kn,X,Y)

print("\nFor Naive-Bayes classifier")
Accu_conf(nb,X,Y)

print("\nFor Decision tree classifier")
Accu_conf(tree,X,Y)
```

```
For K-nearest neighours:
Accuracy: 91.8918918918919
Confusion matrix:
[[15  0  0]
 [ 0 10  1]
 [ 0  2  9]]
Classification report:
 {'setosa': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 15}, 'versicolor': {'precision': 0.83333333
33333334, 'recall': 0.9090909090909091, 'f1-score': 0.8695652173913043, 'support': 11}, 'virginica': {'precision': 0.
9, 'recall': 0.8181818181818182, 'f1-score': 0.8571428571428572, 'support': 11}, 'accuracy': 0.918918918918919, 'macr
o avg': {'precision': 0.9111111111111111, 'recall': 0.9090909090909092, 'f1-score': 0.9089026915113871, 'support': 3
7}, 'weighted avg': {'precision': 0.9207207207207209, 'recall': 0.918918918918919, 'f1-score': 0.9187510491858317, 's
upport': 37}}

For Naive-Bayes classifier
Accuracy: 86.48648648648648
Confusion matrix:
[[10  0  0]
 [ 0 11  1]
 [ 0  4 11]]
Classification report:
 {'setosa': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 10}, 'versicolor': {'precision': 0.73333333
33333333, 'recall': 0.9166666666666666, 'f1-score': 0.8148148148148148, 'support': 12}, 'virginica': {'precision': 0.
9166666666666666, 'recall': 0.7333333333333333, 'f1-score': 0.8148148148148148, 'support': 15}, 'accuracy': 0.8648648
648648649, 'macro avg': {'precision': 0.8833333333333333, 'recall': 0.8833333333333333, 'f1-score': 0.876543209876543
2, 'support': 37}, 'weighted avg': {'precision': 0.8797297297297296, 'recall': 0.8648648648648649, 'f1-score': 0.8648
648648648649, 'support': 37}}

For Decision tree classifier
Accuracy: 97.2972972972973
Confusion matrix:
[[11  0  0]
 [ 0 13  1]
 [ 0  0 12]]
Classification report:
 {'setosa': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 11}, 'versicolor': {'precision': 1.0, 'reca
ll': 0.9285714285714286, 'f1-score': 0.962962962962963, 'support': 14}, 'virginica': {'precision': 0.923076923076923
1, 'recall': 1.0, 'f1-score': 0.9600000000000001, 'support': 12}, 'accuracy': 0.972972972972973, 'macro avg': {'preci
sion': 0.9743589743589745, 'recall': 0.9761904761904763, 'f1-score': 0.9743209876543211, 'support': 37}, 'weighted av
g': {'precision': 0.9750519750519752, 'recall': 0.972972972972973, 'f1-score': 0.9730130130130131, 'support': 37}}
```

In [18]:
```
def k_fold_acc(clf,X,Y):
  score = cross_val_score(clf,X,Y,cv=10)
  return score.mean()*100
```

In [19]:
```
import matplotlib.pyplot as plt

hist_acc=[]
hist_acc.append(k_fold_acc(tree,X,Y))
hist_acc.append(k_fold_acc(kn,X,Y))
hist_acc.append(k_fold_acc(nb,X,Y))

plt.ylim(90,100)
plt.bar(["Decision tree","K-nearest neighour","Naive bayes"],hist_acc)
```

Out[19]: <BarContainer object of 3 artists>



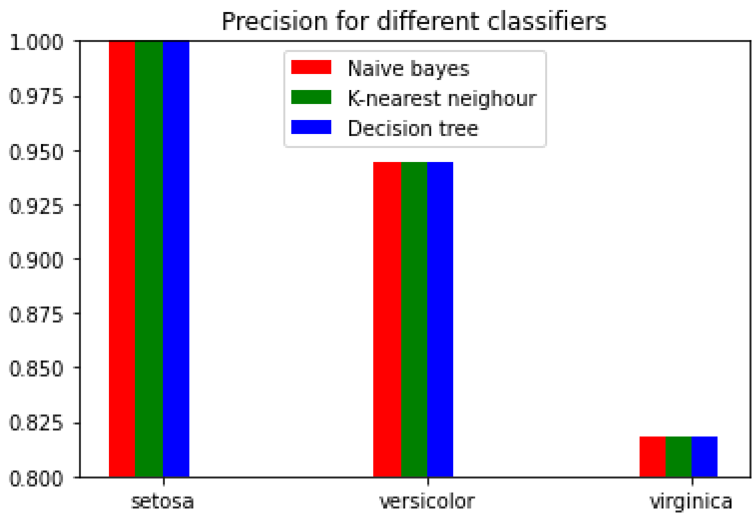
In [20]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
report = []
tree.fit(X_train,Y_train)
Y_pred = tree.predict(X_test)

report.append(classification_report(Y_test,Y_pred,target_names=Y.unique(), output_dict=True))

kn.fit(X_train,Y_train)
Y_pred = kn.predict(X_test)

report.append(classification_report(Y_test,Y_pred,target_names=Y.unique(), output_dict=True))

nb.fit(X_train,Y_train)
Y_pred = nb.predict(X_test)

report.append(classification_report(Y_test,Y_pred,target_names=Y.unique(), output_dict=True))
report
```

Out[20]:
```
[{'accuracy': 0.918918918918919,
  'macro avg': {'f1-score': 0.9253539253539254,
   'precision': 0.9208754208754208,
   'recall': 0.9315789473684211,
   'support': 37},
  'setosa': {'f1-score': 1.0, 'precision': 1.0, 'recall': 1.0, 'support': 8},
  'versicolor': {'f1-score': 0.918918918918919,
   'precision': 0.9444444444444444,
   'recall': 0.8947368421052632,
   'support': 19},
  'virginica': {'f1-score': 0.8571428571428572,
   'precision': 0.8181818181818182,
   'recall': 0.9,
   'support': 10},
  'weighted avg': {'f1-score': 0.9197537305645416,
   'precision': 0.9223314223314223,
   'recall': 0.918918918918919,
   'support': 37}},
 {'accuracy': 0.918918918918919,
  'macro avg': {'f1-score': 0.9253539253539254,
   'precision': 0.9208754208754208,
   'recall': 0.9315789473684211,
   'support': 37},
  'setosa': {'f1-score': 1.0, 'precision': 1.0, 'recall': 1.0, 'support': 8},
  'versicolor': {'f1-score': 0.918918918918919,
   'precision': 0.9444444444444444,
   'recall': 0.8947368421052632,
   'support': 19},
  'virginica': {'f1-score': 0.8571428571428572,
   'precision': 0.8181818181818182,
   'recall': 0.9,
   'support': 10},
  'weighted avg': {'f1-score': 0.9197537305645416,
   'precision': 0.9223314223314223,
   'recall': 0.918918918918919,
   'support': 37}},
 {'accuracy': 0.918918918918919,
  'macro avg': {'f1-score': 0.9253539253539254,
   'precision': 0.9208754208754208,
   'recall': 0.9315789473684211,
   'support': 37},
```

```
 'setosa': {'f1-score': 1.0, 'precision': 1.0, 'recall': 1.0, 'support': 8},
 'versicolor': {'f1-score': 0.918918918918919,
  'precision': 0.9444444444444444,
  'recall': 0.8947368421052632,
  'support': 19},
 'virginica': {'f1-score': 0.8571428571428572,
  'precision': 0.8181818181818182,
  'recall': 0.9,
  'support': 10},
 'weighted avg': {'f1-score': 0.9197537305645416,
  'precision': 0.9223314223314223,
  'recall': 0.918918918918919,
  'support': 37}}]
```

In [21]:
```
cat = ['setosa','versicolor','virginica']
tree_prec=[]
kn_prec=[]
nb_prec=[]

for i in cat:
 tree_prec.append(report[0][i]['precision'])
 kn_prec.append(report[1][i]['precision'])
 nb_prec.append(report[2][i]['precision'])



plt.bar(cat,nb_prec,color='r',width=-0.2,align='edge')
plt.bar(cat,kn_prec,color='g',width=0.2,align='center')
plt.bar(cat,tree_prec,color='b',width=0.1,align='edge')
plt.ylim(0.8,1)
plt.legend(['Naive bayes','K-nearest neighour','Decision tree'], loc ="upper center")
plt.title('Precision for different classifiers')
```

Out[21]:
```
Text(0.5, 1.0, 'Precision for different classifiers')
```



In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
!cp  "/content/drive/MyDrive/Colab Notebooks/Practical 5.ipynb" ./
!jupyter nbconvert --to html "Practical 5.ipynb"
```