ASSIGNMENT 2

Kulsoom Khurshid

SP20-BCS-044

TASK 1

This dataset is related to flowers. There are four characteristics i.e., sepal length, sepal width, petal length and petal width. On the basis of these characteristics classification is made whether the flower is Setosa, Versicolor and Virginica. This dataset is divided in two, half data is used to train the model whereas the other half is used for testing to check if the flower is correctly categorized.

import seaborn as sns

This library that I have used to load the Iris dataset.

The two algorithms that we have used are Logistic regression and Decision Tree.

```
In [28]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import train_test_split
         import seaborn as sns
         import os
         os.getcwd()
Out[28]: 'D:\\Machine Learning'
```

```
In [8]: # We are reading our data
        df=sns.load_dataset("iris")
        df.head()
Out[8]:
```

	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

TASK 2

df.describe()

```
In [2]: # We are reading our data
        df=sns.load_dataset("iris")
        df.describe()
Out[2]:
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

df.info()

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

TASK 3

Logistic Regression

```
In [10]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import train_test_split
         import seaborn as sns
         import os
         os.getcwd()

Out[10]: 'D:\\Machine Learning'
```

```
In [11]: # We are reading our data
         df=sns.load_dataset("iris")
         df.head()
```

Out[11]:

	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

```
In [12]: from sklearn.model_selection import train_test_split
         X_train, X_test, Y_train, Y_test = train_test_split(df.iloc[:,:-1], df.iloc[:,-1], test_size=0.2, random_state=2)

In [13]: from sklearn.linear_model import LogisticRegression

In [14]: clf1=LogisticRegression()

In [15]: clf1.fit(X_train, Y_train)

Out[15]: LogisticRegression()

In [16]: y_pred1 = clf1.predict(X_test)

In [19]: from sklearn.metrics import accuracy_score,confusion_matrix
         print("Accuracy of Logistic Regression",accuracy_score(Y_test,y_pred1))

         Accuracy of Logistic Regression 0.9666666666666667

In [20]: confusion_matrix(Y_test,y_pred1)

Out[20]: array([[14,  0,  0],
                [ 0,  7,  1],
                [ 0,  0,  8]], dtype=int64)
```

In [21]:
```
result = pd.DataFrame()
result['Actual Label'] = Y_test
result['Logistic Regression Prediction'] = y_pred1
```

In [22]:
```
result.sample(10)
```

Out[22]:

	Actual Label	Logistic Regression Prediction
74	versicolor	versicolor
115	virginica	virginica
113	virginica	virginica
29	setosa	setosa
87	versicolor	versicolor
127	virginica	virginica
6	setosa	setosa
77	versicolor	virginica
25	setosa	setosa
128	virginica	virginica

In [23]:
```
from sklearn.metrics import recall_score,precision_score,f1_score
```

In [24]:
```
print("For Logistic regression Model")
print("-"*50)
cdf = pd.DataFrame(confusion_matrix(Y_test,y_pred1),columns=list(range(0,3)))
print(cdf)
print("-"*50)
print("Precision - ",precision_score(Y_test,y_pred1,average='macro'))
print("Recall - ",recall_score(Y_test,y_pred1,average='macro'))
print("F1 score - ",f1_score(Y_test,y_pred1,average='macro'))
```

```
For Logistic regression Model
--------------------------------------------------
    0  1  2
0  14  0  0
1   0  7  1
2   0  0  8
--------------------------------------------------
Precision -  0.9629629629629629
Recall -  0.9583333333333334
F1 score -  0.9581699346405229
```

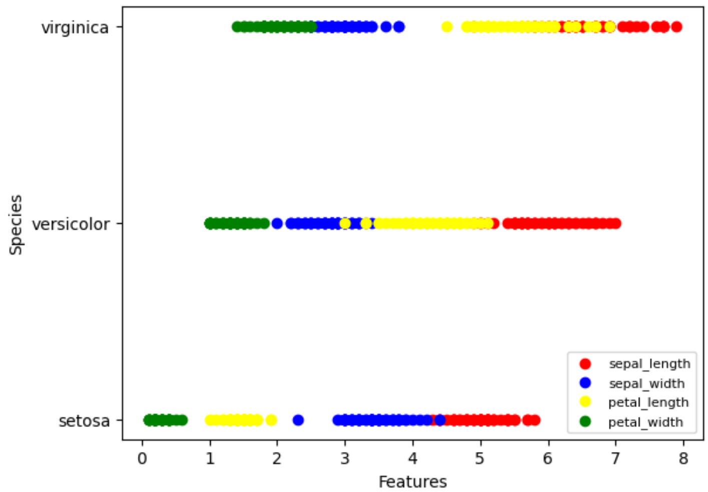
In [25]:
```
precision_score(Y_test,y_pred1,average='macro')
```

Out[25]: 0.9629629629629629

In [26]:
```
from sklearn.metrics import classification_report
print (classification_report(Y_test, y_pred1))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       1.00      0.88      0.93         8
   virginica       0.89      1.00      0.94         8

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

In [28]:
```
plt.xlabel('Features')
plt.ylabel('Species')

pltX = df.loc[:,'sepal_length']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='red', label='sepal_length')

pltX = df.loc[:,'sepal_width']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='blue', label='sepal_width')

pltX = df.loc[:,'petal_length']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='yellow', label='petal_length')

pltX = df.loc[:,'petal_width']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='green', label='petal_width')

plt.legend(loc=4, prop={'size':8})
plt.show()
```



TASK 4

Decision Tree

In [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
import os
os.getcwd()
from sklearn.datasets import make_classification
```

In [19]:
```
x,y= make_classification(n_features=4, n_redundant=0, n_informative=4, n_clusters_per_class=1)
```

In [20]:
```
# We are reading our data
df=sns.load_dataset("iris")
df.head()
```

Out[20]:

	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

In [21]:
```
# function for row sampling
def sample_rows(df, percent):
    return df.sample(int(percent*df.shape[0]))
```

In [22]:
```
#function for column sampling
def sample_features(df, percent):
    cols= random.sample(df.columns.tolist()[:-1], int(percent*df.shape[1]))
    return df[cols]
```

In [23]:
```
#function for combined sampling
def combined_sampling(df, row_percent, col_percent):
    new_df=sample_rows(df, row_percent)
    return sample_features(new_df, col_percent)
```

In [24]:
```
df1=sample_rows(df, 0.1)
```

In [25]:
```
df2=sample_rows(df, 0.1)
```

In [26]:
```
df3= sample_rows(df, 0.1)
```

In [27]:
```
df3.shape
```

Out[27]: (15, 5)

In [28]:
```
from sklearn.tree import DecisionTreeClassifier
clf1= DecisionTreeClassifier()
clf2= DecisionTreeClassifier()
clf3= DecisionTreeClassifier()
```

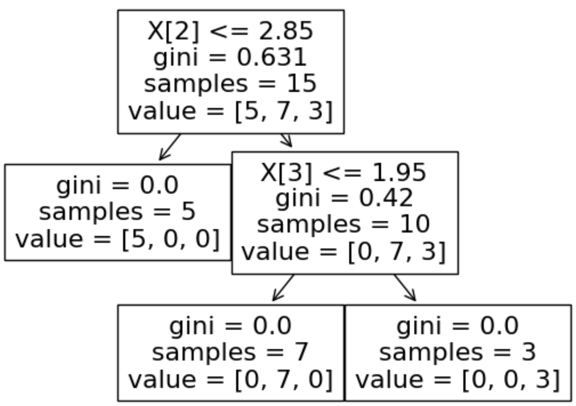
In [32]:
```
clf1.fit(df1.iloc[:,0:4],df1.iloc[:,-1])
clf2.fit(df2.iloc[:,0:4],df1.iloc[:,-1])
clf3.fit(df3.iloc[:,0:4],df1.iloc[:,-1])
```

Out[32]: DecisionTreeClassifier()

In [33]:
```
from sklearn.tree import plot_tree
```

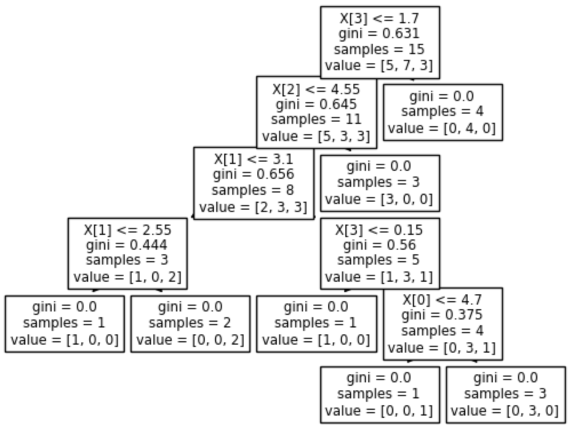
In [34]:
```
plot_tree(clf1)
```

Out[34]:
```
[Text(0.4, 0.8333333333333334, 'X[2] <= 2.85\ngini = 0.631\nsamples = 15\nvalue = [5, 7, 3]'),
 Text(0.2, 0.5, 'gini = 0.0\nsamples = 5\nvalue = [5, 0, 0]'),
 Text(0.6, 0.5, 'X[3] <= 1.95\ngini = 0.42\nsamples = 10\nvalue = [0, 7, 3]'),
 Text(0.4, 0.16666666666666666, 'gini = 0.0\nsamples = 7\nvalue = [0, 7, 0]'),
 Text(0.8, 0.16666666666666666, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]')]
```



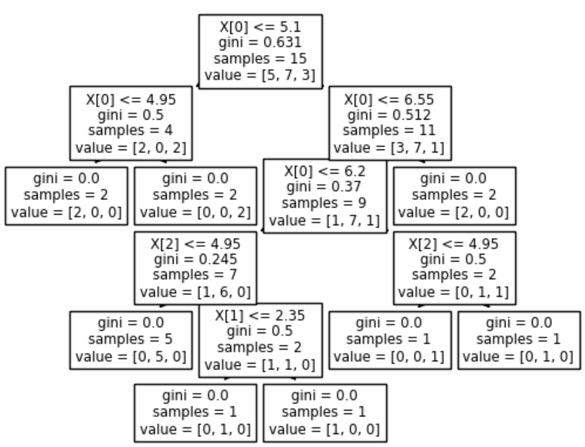
In [35]:
```python
plot_tree(clf2)
```

Out[35]:
```
[Text(0.6666666666666666, 0.9166666666666666, 'X[3] <= 1.7\ngini = 0.631\nsamples = 15\nvalue = [5, 7, 3]'),
 Text(0.5555555555555556, 0.75, 'X[2] <= 4.55\ngini = 0.645\nsamples = 11\nvalue = [5, 3, 3]'),
 Text(0.4444444444444444, 0.5833333333333334, 'X[1] <= 3.1\ngini = 0.656\nsamples = 8\nvalue = [2, 3, 3]'),
 Text(0.2222222222222222, 0.4166666666666667, 'X[1] <= 2.55\ngini = 0.444\nsamples = 3\nvalue = [1, 0, 2]'),
 Text(0.1111111111111111, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0]'),
 Text(0.3333333333333333, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.6666666666666666, 0.4166666666666667, 'X[3] <= 0.15\ngini = 0.56\nsamples = 5\nvalue = [1, 3, 1]'),
 Text(0.5555555555555556, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0]'),
 Text(0.7777777777777778, 0.25, 'X[0] <= 4.7\ngini = 0.375\nsamples = 4\nvalue = [0, 3, 1]'),
 Text(0.6666666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.8888888888888888, 0.08333333333333333, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0]'),
 Text(0.6666666666666666, 0.5833333333333334, 'gini = 0.0\nsamples = 3\nvalue = [3, 0, 0]'),
 Text(0.7777777777777778, 0.75, 'gini = 0.0\nsamples = 4\nvalue = [0, 4, 0]')]
```



In [36]:
```python
plot_tree(clf3)
```

Out[36]:
```
[Text(0.4444444444444444, 0.9166666666666666, 'X[0] <= 5.1\ngini = 0.631\nsamples = 15\nvalue = [5, 7, 3]'),
 Text(0.2222222222222222, 0.75, 'X[0] <= 4.95\ngini = 0.5\nsamples = 4\nvalue = [2, 0, 2]'),
 Text(0.1111111111111111, 0.5833333333333334, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0]'),
 Text(0.3333333333333333, 0.5833333333333334, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.6666666666666666, 0.75, 'X[0] <= 6.55\ngini = 0.512\nsamples = 11\nvalue = [3, 7, 1]'),
 Text(0.5555555555555556, 0.5833333333333334, 'X[0] <= 6.2\ngini = 0.37\nsamples = 9\nvalue = [1, 7, 1]'),
 Text(0.3333333333333333, 0.4166666666666667, 'X[2] <= 4.95\ngini = 0.245\nsamples = 7\nvalue = [1, 6, 0]'),
 Text(0.2222222222222222, 0.25, 'gini = 0.0\nsamples = 5\nvalue = [0, 5, 0]'),
 Text(0.4444444444444444, 0.25, 'X[1] <= 2.35\ngini = 0.5\nsamples = 2\nvalue = [1, 1, 0]'),
 Text(0.3333333333333333, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.5555555555555556, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0]'),
 Text(0.7777777777777778, 0.4166666666666667, 'X[2] <= 4.95\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(0.6666666666666666, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.8888888888888888, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.7777777777777778, 0.5833333333333334, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0]')]
```



TASK 5

In [78]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy of Logistic Regression",accuracy_score(Y_test,y_pred1))
print("Accuracy of Decision Trees",accuracy_score(Y_test,y_pred2))
```

```
Accuracy of Logistic Regression 0.9666666666666667
Accuracy of Decision Trees 0.9333333333333333
```

In [80]:
```
print("Logistic Regression Confusion Matrix\n")
pd.DataFrame(confusion_matrix(Y_test,y_pred1),columns=list(range(0,3)))
```

```
Logistic Regression Confusion Matrix
```

Out[80]:

	0	1	2
0	14	0	0
1	0	7	1
2	0	0	8

In [81]:
```
print("Decision Tree Confusion Matrix\n")
pd.DataFrame(confusion_matrix(Y_test,y_pred2),columns=list(range(0,3)))
```

```
Decision Tree Confusion Matrix
```

Out[81]:

	0	1	2
0	14	0	0
1	0	7	1
2	0	1	7

In [85]:
```
print("For Logistic regression Model")
print("-"*50)
cdf = pd.DataFrame(confusion_matrix(Y_test,y_pred1),columns=list(range(0,3)))
print(cdf)
print("-"*50)
print("Precision - ",precision_score(Y_test,y_pred1,average='macro'))
print("Recall - ",recall_score(Y_test,y_pred1,average='macro'))
print("F1 score - ",f1_score(Y_test,y_pred1,average='macro'))
```

```
For Logistic regression Model
--------------------------------------------------
    0  1  2
0  14  0  0
1   0  7  1
2   0  0  8
--------------------------------------------------
Precision -  0.9629629629629629
Recall -  0.9583333333333334
F1 score -  0.9581699346405229
```

In [86]:
```
print("For DT Model")
print("-"*50)
cdf = pd.DataFrame(confusion_matrix(Y_test,y_pred2),columns=list(range(0,3)))
print(cdf)
print("-"*50)
print("Precision - ",precision_score(Y_test,y_pred2,average='macro'))
print("Recall - ",recall_score(Y_test,y_pred2,average='macro'))
print("F1 score - ",f1_score(Y_test,y_pred2,average='macro'))
```

```
For DT Model
--------------------------------------------------
    0  1  2
0  14  0  0
1   0  7  1
2   0  1  7
--------------------------------------------------
Precision -  0.9166666666666666
Recall -  0.9166666666666666
F1 score -  0.9166666666666666
```

In [87]:
```
precision_score(Y_test,y_pred1,average='macro')
```
Out[87]:
```
0.9629629629629629
```

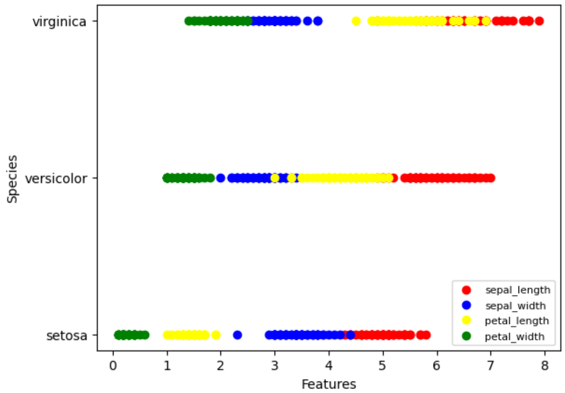
In [88]:
```
precision_score(Y_test,y_pred2,average='macro')
```
Out[88]:
```
0.9166666666666666
```

In [89]:
```
recall_score(Y_test,y_pred2,average=None)
```
Out[89]:
```
array([1.   , 0.875, 0.875])
```

In [90]:
```
from sklearn.metrics import classification_report
print (classification_report(Y_test, y_pred1))
```
```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       1.00      0.88      0.93         8
   virginica       0.89      1.00      0.94         8

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

In [93]:
```
from sklearn.metrics import classification_report
print (classification_report(Y_test, y_pred2))
```
```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       0.88      0.88      0.88         8
   virginica       0.88      0.88      0.88         8

    accuracy                           0.93        30
   macro avg       0.92      0.92      0.92        30
weighted avg       0.93      0.93      0.93        30
```

In [91]:
```
plt.xlabel('Features')
plt.ylabel('Species')

pltX = df.loc[:,'sepal_length']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='red', label='sepal_length')

pltX = df.loc[:,'sepal_width']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='blue', label='sepal_width')

pltX = df.loc[:,'petal_length']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='yellow', label='petal_length')

pltX = df.loc[:,'petal_width']
pltY = df.loc[:, 'species']
plt.scatter(pltX, pltY, color='green', label='petal_width')

plt.legend(loc=4, prop={'size':8})
plt.show()
```



In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)

# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend
() is called with no argument.
```