PRACTICAL - 8

AIM: Write a program to classify IRIS data using Random forest classifier

- ## Importing necessary libraries

In [1]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

- ## Loading the dataset

In [2]:
```
df=pd.read_csv("iris_code.csv")
df
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

- ## Display dataset information

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Id              150 non-null    int64  
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

- ## Summary statistics of the dataset

In [4]:
```
df.describe()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

- **List column names of the dataset**

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [6]:
```
x=df.groupby(df['Species'])
x.groups.keys()
```

Out[6]:
```
dict_keys(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
```

In [7]:
```
features = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
features
```

Out[7]:

	Sepal engthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

• Group dataset by species

In [8]:
```
target=df["Species"]
target
```

Out[8]:
```
0          Iris-setosa
1          Iris-setosa
2          Iris-setosa
3          Iris-setosa
4          Iris-setosa
              ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

• Working with LogisticRegression

In [9]:
```
# Initialize Rondom Forest Classification model
rf=RandomForestClassifier()
```

In [10]:
```
# Train the model on the full dataset
rf.fit(features,target)
```

Out[10]:
```
▾  RandomForest lassifier
RandomForestClassifier()
```

In [11]:
```
# Make a prediction with example input
x=rf.predict([[1,1,1,1]])[0]
print(x)
```

```
Iris-versicolor
C:\Users\PARAM\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X do es not have valid feature names, but RandomForestClassifier was fitted with featu re names
  warnings.warn(
```

- **Using train_test_split()**

In [12]:
```
# Split dataset into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(features,target,test_size=0.30,ra
```

In [13]:
```
X_train
```

Out[13]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
81	5.5	2.4	3.7	1.0
133	6.3	2.8	5.1	1.5
137	6.4	3.1	5.5	1.8
75	6.6	3.0	4.4	1.4
109	7.2	3.6	6.1	2.5
...	...	...	...	...
71	6.1	2.8	4.0	1.3
106	4.9	2.5	4.5	1.7
14	5.8	4.0	1.2	0.2
92	5.8	2.6	4.0	1.2
102	7.1	3.0	5.9	2.1

105 rows × 4 columns

In [14]:
```
rf.fit(X_train,y_train)
```

Out[14]:
```
RandomForest lassifier
RandomForestClassifier()
```

In [15]:
```
y_pred=rf.predict(X_test)
print(y_pred)
```

```
['Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa']
```

In [16]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[16]: 1.0

In [18]:
```
sol=accuracy_score(y_test,y_pred)
```

In [19]:
```
print(sol)
```

```
1.0
```

In [20]:
```
data=pd.DataFrame({'Actual':y_test,'Predicted': y_pred})
data
```

Out[20]:

	Actual	Predicted
73	Iris-versicolor	Iris-versicolor
18	Iris-setosa	Iris-setosa
118	Iris-virginica	Iris-virginica
78	Iris-versicolor	Iris-versicolor
76	Iris-versicolor	Iris-versicolor
31	Iris-setosa	Iris-setosa
64	Iris-versicolor	Iris-versicolor
141	Iris-virginica	Iris-virginica
68	Iris-versicolor	Iris-versicolor
82	Iris-versicolor	Iris-versicolor
110	Iris-virginica	Iris-virginica
12	Iris-setosa	Iris-setosa
36	Iris-setosa	Iris-setosa
9	Iris-setosa	Iris-setosa
19	Iris-setosa	Iris-setosa
56	Iris-versicolor	Iris-versicolor
104	Iris-virginica	Iris-virginica
69	Iris-versicolor	Iris-versicolor
55	Iris-versicolor	Iris-versicolor
132	Iris-virginica	Iris-virginica
29	Iris-setosa	Iris-setosa
127	Iris-virginica	Iris-virginica
26	Iris-setosa	Iris-setosa
128	Iris-virginica	Iris-virginica
131	Iris-virginica	Iris-virginica
145	Iris-virginica	Iris-virginica
108	Iris-virginica	Iris-virginica
143	Iris-virginica	Iris-virginica
45	Iris-setosa	Iris-setosa
30	Iris-setosa	Iris-setosa
22	Iris-setosa	Iris-setosa

15	Iris-setosa	Iris-setosa
65	Iris-versicolor	Iris-versicolor

	Actual	Predicted
11	Iris-setosa	Iris-setosa
42	Iris-setosa	Iris-setosa
146	Iris-virginica	Iris-virginica
51	Iris-versicolor	Iris-versicolor
27	Iris-setosa	Iris-setosa
4	Iris-setosa	Iris-setosa
32	Iris-setosa	Iris-setosa
142	Iris-virginica	Iris-virginica
85	Iris-versicolor	Iris-versicolor
86	Iris-versicolor	Iris-versicolor
16	Iris-setosa	Iris-setosa
10	Iris-setosa	Iris-setosa

• Wroking with other Libraries

In [21]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [22]:
```
svm=svm.SVC()
knn=KNeighborsClassifier()
dt=DecisionTreeClassifier()
lr=LogisticRegression()
gbc=GradientBoostingClassifier()
```

In [23]:
```
svm.fit(X_train,y_train)
knn.fit(X_train,y_train)
dt.fit(X_train,y_train)
lr.fit(X_train,y_train)
gbc.fit(X_train,y_train)
```

Out[23]:
```
GradientBoostingClassifier()
```

In [24]:
```
y_pred1 = svm.predict(X_test)
y_pred2= knn.predict(X_test)
y_pred3 = dt.predict(X_test)
y_pred4 = lr.predict(X_test)
y_pred5 = gbc.predict(X_test)
```

In [25]:
```
s1=accuracy_score(y_test,y_pred1)
s2=accuracy_score(y_test,y_pred2)
s5=accuracy_score(y_test,y_pred5)
s6=accuracy_score(y_test,y_pred)
```

In [26]:
```
print(s1)
print(s2)
print(s3)
print(s4)
print(s5)
```

```
1.0
1.0
1.0
1.0
1.0
```

In [27]:
```
l=[s1,s2,s3,s4,s5,s6]
l
```

Out[27]:
```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

In [28]:
```
import matplotlib.pyplot as plt
```

In [29]:
```
model=['SVM','KNN','DT','RF','GBC','LR']
colors=['b','g','r','c','m','y']
plt.bar(model, l,width=0.5,color=colors)
plt.show()
```