Aim: Build a decision tree model for a given dataset

In [7]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
```

With Iris Dataset

In [8]:
```
df = pd.read_csv("Iris.csv")
```

In [9]:
```
df
```

Out[9]:

	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

In [10]:
```
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
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [12]:
```
X = df.iloc[:,1:5]
Y = df.iloc[:,-1]
```

In [13]:
```
X
```

Out[13]:

	SepalLengthCm	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

In [14]:
```
Y.shape
```

Out[14]:
```
(150,)
```

In [15]:
```
X.shape
```

Out[15]: (150, 4)

In [16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random
```

In [17]:
```
Y_test.shape
```

Out[17]: (50,)

In [18]:
```
X_train.shape
```

Out[18]: (100, 4)

In [19]:
```
Y_train.shape
```

Out[19]: (100,)

In [20]:
```
df.describe()
```

Out[20]:

	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

In [21]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
```

In [22]:
```
dt.fit(X_train,Y_train)
```

Out[22]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [23]:
```
Y_pred = dt.predict(X_test)
```

In [24]:
```
dt.score(X_test,Y_test)
```

Out[24]: 1.0

In [25]:
```
import numpy as np
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
```

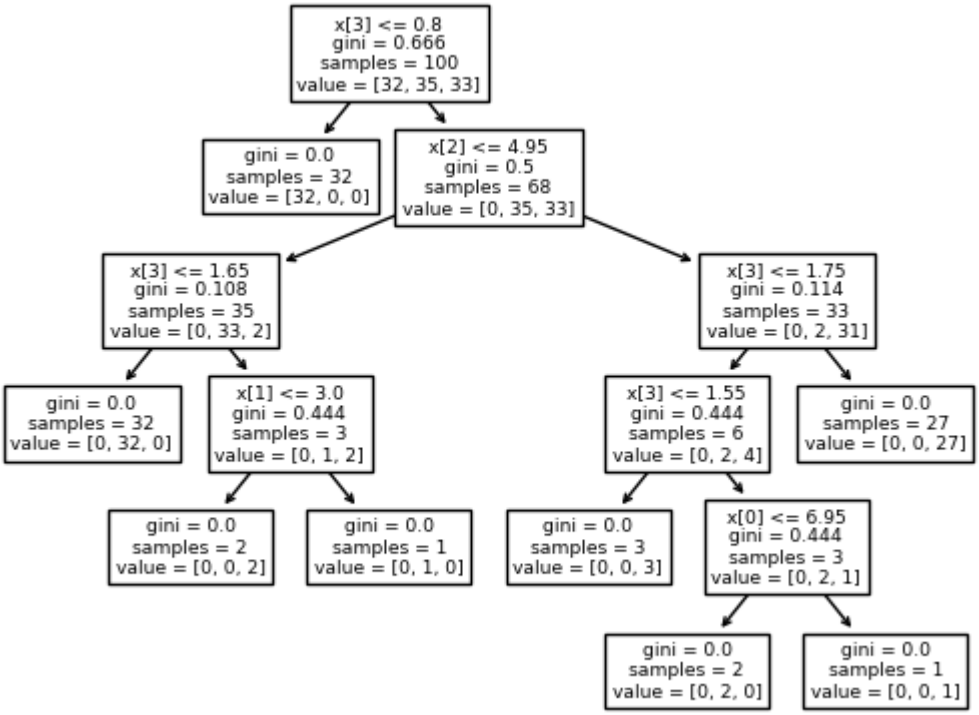
In [26]:
```
print('Accuracy: %.3f' % accuracy_score(Y_test, Y_pred))
print('f1 score: %.3f' % f1_score(Y_test, Y_pred,average='micro'))
print('recall: %.3f' % recall_score(Y_test, Y_pred, average='macro'))
print('Precision: %.3f' % precision_score(Y_test, Y_pred,average='macro'))
```

```
Accuracy: 1.000
f1 score: 1.000
recall: 1.000
Precision: 1.000
```

In [27]:
```
from sklearn import tree
clf=tree.DecisionTreeClassifier()
clf=clf.fit(X_train,Y_train)
```

In [28]:
```
tree.plot_tree(clf)
```

Out[28]:
```
[Text(0.4, 0.9166666666666666, 'x[3] <= 0.8\ngini = 0.666\nsamples = 100\nvalue
= [32, 35, 33]'),
 Text(0.3, 0.75, 'gini = 0.0\nsamples = 32\nvalue = [32, 0, 0]'),
 Text(0.5, 0.75, 'x[2] <= 4.95\ngini = 0.5\nsamples = 68\nvalue = [0, 35, 3
3]'),
 Text(0.2, 0.5833333333333334, 'x[3] <= 1.65\ngini = 0.108\nsamples = 35\nvalue
= [0, 33, 2]'),
 Text(0.1, 0.4166666666666667, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]'),
 Text(0.3, 0.4166666666666667, 'x[1] <= 3.0\ngini = 0.444\nsamples = 3\nvalue =
[0, 1, 2]'),
 Text(0.2, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.4, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8, 0.5833333333333334, 'x[3] <= 1.75\ngini = 0.114\nsamples = 33\nvalue
= [0, 2, 31]'),
 Text(0.7, 0.4166666666666667, 'x[3] <= 1.55\ngini = 0.444\nsamples = 6\nvalue
= [0, 2, 4]'),
 Text(0.6, 0.25, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.8, 0.25, 'x[0] <= 6.95\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(0.7, 0.08333333333333333, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(0.9, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.9, 0.4166666666666667, 'gini = 0.0\nsamples = 27\nvalue = [0, 0, 27]')]
```



In [29]:
```
import numpy as np
user_input = []
SepalLengthCm = 2.3
SepalWidthCm = 5.3
PetalLengthCm = 1.2
PetalWidthCm = 2.6
user_input.append([SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm])

user_input = np.array(user_input)

predicted_classes = dt.predict(user_input)

print(f"predicted class: {predicted_classes}")
```

```
predicted class: ['Iris-versicolor']
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but DecisionTreeC
lassifier was fitted with feature names
  warnings.warn(
```

In [30]:
```
X.describe()
```

Out[30]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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

In [31]:
```
# import numpy as np

user_input = []

SepalLengthCm = float(input("Enter value between 4.300000 to 7.900000 for SepalL
SepalWidthCm = float(input("Enter value between 2.000000 to 4.400000  for SepalW
PetalLengthCm = float(input("Enter value between 1.000000 to 6.900000  for Petal
PetalWidthCm = float(input("Enter value between 0.100000 to 2.500000  for PetalW
user_input.append([SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm])

user_input = np.array(user_input)

predicted_classes = dt.predict(user_input)

print(predicted_classes)
```

```
['Iris-virginica']
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but DecisionTreeC
lassifier was fitted with feature names
  warnings.warn(
```

In [32]:
```
# With Automobile Dataset
```

In [33]:
```
df1 = pd.read_csv("Automobile.csv")
```

In [34]:
```
df1
```

Out[34]:

	name	mpg	cylinders	displacement	horsepower	weight	acceleration	model
0	chevrolet chevelle malibu	18.0	8.0	307.0	130.0	3504.0	12.0	
1	buick skylark 320	15.0	8.0	350.0	165.0	3693.0	11.5	
2	plymouth satellite	18.0	8.0	318.0	150.0	3436.0	11.0	
3	amc rebel sst	16.0	8.0	304.0	150.0	3433.0	12.0	
4	ford torino	17.0	8.0	302.0	140.0	3449.0	10.5	
...	...	...	...	...	...	...	...	
393	ford mustang gl	27.0	4.0	140.0	86.0	2790.0	15.6	
394	vw pickup	44.0	4.0	97.0	52.0	2130.0	24.6	
395	dodge rampage	32.0	4.0	135.0	84.0	2295.0	11.6	
396	ford ranger	28.0	4.0	120.0	79.0	2625.0	18.6	
397	chevy s-10	31.0	4.0	119.0	82.0	2720.0	19.4	

398 rows × 9 columns

In [35]:
```
s=df1['origin']=='usa'
s.sum()
```

Out[35]:
```
249
```

In [36]:
```
df1.isnull().sum()
```

Out[36]:
```
name            0
mpg             1
cylinders       1
displacement    5
horsepower      7
weight          5
acceleration    2
model_year      2
origin          0
dtype: int64
```

In [37]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   name          398 non-null    object 
 1   mpg           397 non-null    float64
 2   cylinders     397 non-null    float64
 3   displacement  393 non-null    float64
 4   horsepower    391 non-null    float64
 5   weight        393 non-null    float64
 6   acceleration  396 non-null    float64
 7   model_year    396 non-null    float64
 8   origin        398 non-null    object 
dtypes: float64(7), object(2)
memory usage: 28.1+ KB
```

In [38]:
```
df1.describe()
```

Out[38]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	m
count	397.000000	397.000000	393.000000	391.000000	393.000000	396.000000	3
mean	23.538539	5.458438	192.393130	104.524297	2949.053435	15.671970	
std	7.811191	1.701577	103.205814	38.525101	851.576054	3.961926	
min	9.000000	3.000000	68.000000	46.000000	15.000000	0.500000	
25%	17.500000	4.000000	105.000000	75.000000	2220.000000	13.775000	
50%	23.000000	4.000000	146.000000	94.000000	2790.000000	15.500000	
75%	29.000000	8.000000	260.000000	127.000000	3570.000000	17.225000	
max	46.600000	8.000000	455.000000	230.000000	5140.000000	70.000000	

In [39]:
```
X = pd.concat([df1.iloc[:,1:5], df1.iloc[:,5:7]], axis=1)
Y = df1.iloc[:,8:9]
```

In [40]:
```
X['mpg'] = X['mpg'].fillna(value=23.538539)
# Replace null values in the 'cylinders' column with a specific value
X['cylinders'] = X['cylinders'].fillna(value=5.458438)

# Replace null values in the 'displacement' column with a different value
X['displacement'] = X['displacement'].fillna(value=192.393130)

X['horsepower'] = X['horsepower'].fillna(value=104.524297)

X['weight'] = X['weight'].fillna(value=192.393130)

X['acceleration'] = X['acceleration'].fillna(value=192.393130)

#Y=Y.fillna(104.524297)
```

In [41]:
```
X.isnull().sum()
```

Out[41]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
dtype: int64
```

In [42]:
```
Y.isnull().sum()
```

Out[42]:
```
origin    0
dtype: int64
```

In [43]:
```
X.shape
```

Out[43]:
```
(398, 6)
```

In [44]:
```
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, random
```

In [45]:
```
X_train.shape
```

Out[45]:
```
(298, 6)
```

In [46]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
```

In [47]:
```
dt.fit(X_train,Y_train)
```

Out[47]:
```
DecisionTreeClassifier()
```

In [48]:
```
Y_pred = dt.predict(X_test)
```

In [49]:
```
dt.score(X_test,Y_test)
```

Out[49]:
```
0.89
```

In [50]:
```
import numpy as np
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
```

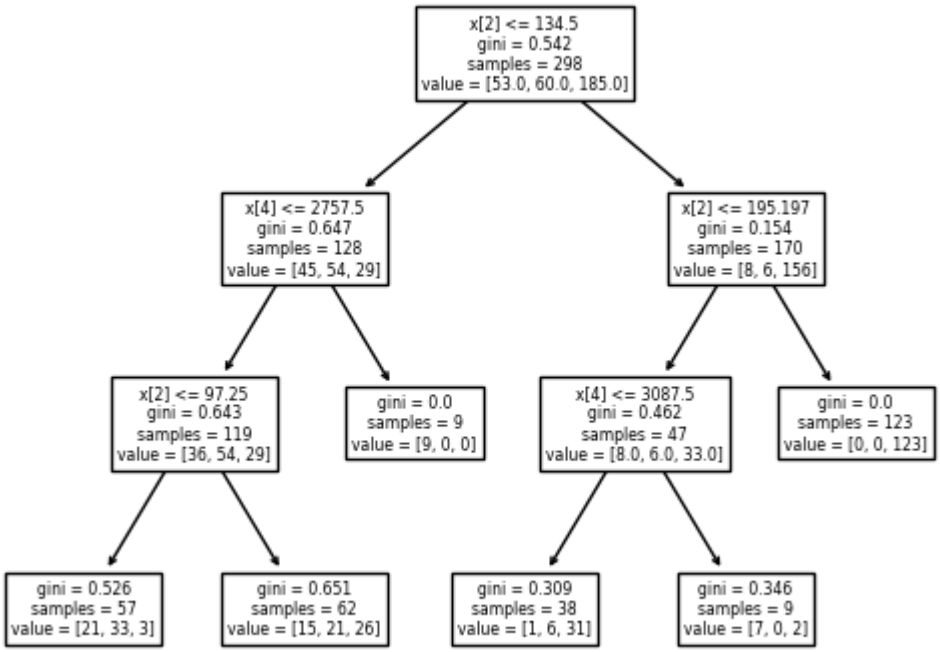
In [51]:
```
print('Accuracy: %.3f' % accuracy_score(Y_test, Y_pred))
print('f1 score: %.3f' % f1_score(Y_test, Y_pred,average='micro'))
print('recall: %.3f' % recall_score(Y_test, Y_pred, average='macro'))
print('Precision: %.3f' % precision_score(Y_test, Y_pred,average='micro'))
```

```
Accuracy: 0.890
f1 score: 0.890
recall: 0.836
Precision: 0.890
```

In [61]:
```
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(X_train,Y_train)
```

```
tree.plot_tree(clf)
```

Out[61]:
```
[Text(0.5555555555555556, 0.875, 'x[2] <= 134.5\ngini = 0.542\nsamples = 298\nv
alue = [53.0, 60.0, 185.0]'),
 Text(0.3333333333333333, 0.625, 'x[4] <= 2757.5\ngini = 0.647\nsamples = 128\n
value = [45, 54, 29]'),
 Text(0.2222222222222222, 0.375, 'x[2] <= 97.25\ngini = 0.643\nsamples = 119\nv
alue = [36, 54, 29]'),
 Text(0.1111111111111111, 0.125, 'gini = 0.526\nsamples = 57\nvalue = [21, 33,
3]'),
 Text(0.3333333333333333, 0.125, 'gini = 0.651\nsamples = 62\nvalue = [15, 21,
26]'),
 Text(0.4444444444444444, 0.375, 'gini = 0.0\nsamples = 9\nvalue = [9, 0, 0]'),
 Text(0.7777777777777778, 0.625, 'x[2] <= 195.197\ngini = 0.154\nsamples = 170
\nvalue = [8, 6, 156]'),
 Text(0.6666666666666666, 0.375, 'x[4] <= 3087.5\ngini = 0.462\nsamples = 47\nv
alue = [8.0, 6.0, 33.0]'),
 Text(0.5555555555555556, 0.125, 'gini = 0.309\nsamples = 38\nvalue = [1, 6, 3
1]'),
 Text(0.7777777777777778, 0.125, 'gini = 0.346\nsamples = 9\nvalue = [7, 0,
2]'),
 Text(0.8888888888888888, 0.375, 'gini = 0.0\nsamples = 123\nvalue = [0, 0, 12
3]')]
```



In [54]:
```
import numpy as np
user_input = []
mpg = 25
cylinders = 5
displacement = 82
horsepower = 50
weight = 200
acceleration = 100
user_input.append([mpg, cylinders, displacement, horsepower, weight, acceleratio

user_input = np.array(user_input)
```

```
predicted_classes = dt.predict(user_input)

print(f"predicted class: {predicted_classes}")
```

```
predicted class: ['europe']
```

```
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but DecisionTreeC
lassifier was fitted with feature names
  warnings.warn(
```

In [55]:
```
X.describe()
```

Out[55]:

	mpg	cylinders	displacement	horsepower	weight	acceleration
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.538539	5.458438	192.393130	104.524297	2914.422024	16.560016
std	7.801347	1.699433	102.553844	38.183948	900.307533	13.120937
min	9.000000	3.000000	68.000000	46.000000	15.000000	0.500000
25%	17.500000	4.000000	105.000000	76.000000	2206.250000	13.800000
50%	23.000000	4.000000	151.000000	95.000000	2764.500000	15.500000
75%	29.000000	8.000000	260.000000	125.000000	3556.000000	17.300000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	192.393130

In [56]:
```
import numpy as np

user_input = []

mpg = float(input("Enter value between 9.000000 to 46.600000 for mpg: "))
cylinders = float(input("Enter value between 3.000000 to 8.000000 for cylinders:
displacement = float(input("Enter value between 68.000000 to 455.000000 for disp
horsepower = float(input("Enter value between 46.000000 to 230.000000 for horsep
weight = float(input("Enter value between 15.000000 to 5140.000000 for weight: "
acceleration = float(input("Enter value between 0.500000 to 192.393130 for accel

user_input.append([mpg, cylinders, displacement, horsepower, weight, acceleratio

user_input = np.array(user_input)

predicted_classes = dt.predict(user_input)

print(predicted_classes)
```

```
['usa']
```

```
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but DecisionTreeC
lassifier was fitted with feature names
  warnings.warn(
```

In [ ]:

In [ ]: