```
import numpy as np #linear algebra
import pandas as pd #data processing
from sklearn.preprocessing import LabelEncoder #if data isstring then

from sklearn.tree import DecisionTreeClassifier
Sidharth_df = pd.read_csv('/content/glass.csv')
Sidharth_df
```

	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.00.1	1
0	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.00	1
1	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.00	1
2	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.00	1
3	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.00	1
4	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.00	0.26	1
...	...	...	...	...	...	...	...	...	...	...
208	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.00	7
209	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.00	7
210	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.00	7
211	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.00	7
212	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.00	7

213 rows × 10 columns

Next steps: View recommended plots

```
#Extracting Features
X = Sidharth_df.iloc[:,:-1]
y= Sidharth_df.iloc[:,9]
#making Model
Sidharth_dt = DecisionTreeClassifier(criterion = "entropy")
Sidharth_dt
DecisionTreeClassifier(criterion='entropy')
#Splitting the dataset into training, testing and predicting values:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 0)


Sidharth_dt.fit(X_train, y_train)
y_pred = Sidharth_dt.predict(X_test)
y_pred
```

```
array([7, 1, 1, 5, 2, 2, 1, 2, 1, 2, 1, 2, 3, 1, 2, 7, 2, 1, 2, 2, 6, 7,
       7, 7, 2, 2, 5, 1, 2, 1, 1, 1, 2, 2, 2, 1, 1, 3, 2, 7, 2, 6, 2, 2,
       1, 2, 1, 2, 1, 2, 2, 7, 7, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2])
```

```
#Making Confusion Matrix
from sklearn.metrics import confusion_matrix
Sidharth_cm = confusion_matrix(y_test, y_pred)
print(Sidharth_cm)
```

```
[[15  5  0  0  0  0]
 [ 5 19  0  0  0  1]
 [ 2  4  2  0  0  0]
 [ 0  0  0  2  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  7]]
```

```
#finding accuracy from the confusion matrix.
a = Sidharth_cm.shape
corrPred = 0
falsePred = 0
for row in range(a[0]):
  for c in range(a[1]):
    if row == c:
      corrPred +=Sidharth_cm[row,c]
    else:
      falsePred += Sidharth_cm[row,c]
print('Correct predictions: ', corrPred)
print('False predictions', falsePred)
print ('\n\nAccuracy of the ID3 is: ', corrPred/(
Sidharth_cm.sum()))
```

```
Correct predictions:  47
False predictions 17


Accuracy of the ID3 is:  0.734375
```

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
Sidharth_iris = load_iris()


decision_tree = DecisionTreeClassifier(random_state=0, max_depth=2)
decision_tree = decision_tree.fit(Sidharth_iris.data,Sidharth_iris.target)


r = export_text(decision_tree,
feature_names=Sidharth_iris['feature_names'])
print(r)
```

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- class: 2
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.datasets import load_iris
Sidharth_iris = load_iris()



X = Sidharth_iris.data
y = Sidharth_iris.target
Sidharth_dataset = pd.read_csv('/content/golf-dataset.csv')
Sidharth_dataset.head()
```

	Outlook	Temp	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes

Next steps: View recommended plots

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.4, random_state=1)



from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
GaussianNB()
```

```
▾ GaussianNB
GaussianNB()
```

```
# making predictions on the testing set
y_pred = gnb.predict(X_test)
print(y_pred)
```

```
[6 7 5 2 1 1 3 3 1 1 3 1 1 2 1 1 3 2 1 1 3 6 3 3 3 1 1 3 3 6 2 6 3 1 1 1 3
 1 2 3 1 7 7 1 7 1 3 1 1 7 1 3 2 1 5 1 7 1 5 7 1 2 3 3 1 1 2 1 7 2 1 2 5 6
 3 3 1 1 1 7 7 3 2 1 7 1]
```

```
y_compare = np.vstack((y_test,y_pred)).T
y_compare[:5,:]
```

```
array([[2, 6],
       [7, 7],
       [2, 5],
       [2, 2],
       [1, 1]])
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

a = cm.shape
corrPred = 0
falsePred = 0
```

```
[[21  2  8  0  1  0]
 [ 9  6  8  2  1  0]
 [ 4  0  4  0  1  0]
 [ 0  3  0  1  0  0]
 [ 0  0  0  0  2  1]
 [ 1  0  0  1  0 10]]
```

```
for row in range(a[0]):
  for c in range(a[1]):
    if row == c:
      corrPred +=cm[row,c]
    else:
      falsePred += cm[row,c]
print('Correct predictions: ', corrPred)
print('False predictions', falsePred)
print ('\n\nAccuracy of the Naive Bayes Clasification is: ',
corrPred/(cm.sum()))
```

```
Correct predictions:  44
False predictions 42


Accuracy of the Naive Bayes Clasification is:  0.5116279069767442
```