```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

```

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import tree
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.tree import plot_tree
from sklearn.metrics import classification_report
import graphviz
```

```
train=pd.read_csv("/content/iris.csv")
train.head()
```

	sepallength	sepalwidth	petallength	petalwidth	class1
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
train.describe()
```

	sepallength	sepalwidth	petallength	petalwidth
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

```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
```

```
5
6 # Encode labels in column 'species'.
7 train['class1']= label_encoder.fit_transform(train['class1'])
8
9 train['class1'].unique()
10 train.head()
```

	sepallength	sepalwidth	petallength	petalwidth	class1
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
1 a = np.array(train)
2 X=train.drop(columns=['class1'])
3 Y=train['class1']
```

```
1 from sklearn.svm import SVC
2 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.9, random_state =1)
3 model = SVC(kernel='linear')
4 model=model.fit(X_train, y_train)
5 y_pred = model.predict(X_test)
6 print("Predicted values:")
7 print(y_pred)
8 print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
9 print("Accuracy : ",accuracy_score(y_test,y_pred)*100)
10 print("Report : ",classification_report(y_test, y_pred))
11 # fitting x samples and y classes
12 model.fit(X, Y)
13
14 model.predict([[1,1,1,1]])
15
```

```
Predicted values:
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 1 2 1 2 0 0 0 1 0 0 2 2 2 2 1 1 2 1 0 2 1 0 0 2 0 2 2 1 1 2 2 0
 1 1 2 1 2 1 0 0 0 2 0 2 2 2 0 0 1 0 2 1 2 2 1 2 2 1 0 1 0 1 1 0 1 0 0 2 2
 2 0 0 2 0 2 0 2 1 0 2 0 1 0 1 1 0 0 1 0 1 1 0 1]
Confusion Matrix:  [[47  0  0]
 [ 0 42  2]
 [ 0  4 40]]
Accuracy :  95.55555555555556
Report :                precision    recall  f1-score   support

           0       1.00      1.00      1.00        47
           1       0.91      0.95      0.93        44
           2       0.95      0.91      0.93        44

    accuracy                           0.96       135
   macro avg       0.96      0.95      0.95       135
weighted avg       0.96      0.96      0.96       135
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
array([0])
```

```
1 model.predict([[5,6,7,8]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
array([2])
```