In [1]:

```
import pandas as pd
```

In [2]:

```
dataset = pd.read_csv('IRIS.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

In [3]:

```
y
```

Out[3]:

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica'], dtype=object)
```

In [4]:

```
#Convert Categorical variable 'y' into numerical value

from sklearn.preprocessing import LabelEncoder
labelEncoder = LabelEncoder()
y = labelEncoder.fit_transform(y)
```

In [5]:

```
y
```

Out[5]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [6]:

```
#Splitting training and testing data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_stat
e = 0)
```

In [7]:

```
scores = []
```

In [8]:

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
C:\Users\Priyanshi\Anaconda3\lib\site-packages\sklearn\linear_model\logist
ic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Priyanshi\Anaconda3\lib\site-packages\sklearn\linear_model\logist
ic.py:469: FutureWarning: Default multi_class will be changed to 'auto' in
0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[8]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [9]:

```
#Prediction

y_pred = classifier.predict(X_test)
```

In [10]:

```
y_test
```

Out[10]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [11]:

```
y_pred
```

Out[11]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 2,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [12]:

```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [13]:

```
cm
```

Out[13]:

```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]], dtype=int64)
```

In [14]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [15]:

```
scores.append(('Logistic Regression', score))
```

In [16]:

```
scores
```

Out[16]:

```
[('Logistic Regression', 0.9666666666666667)]
```

In [17]:

```
# K-Nearest Neighbours
from sklearn.neighbors import KNeighborsClassifier
classsifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

```
C:\Users\Priyanshi\Anaconda3\lib\site-packages\sklearn\linear_model\logist
ic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Priyanshi\Anaconda3\lib\site-packages\sklearn\linear_model\logist
ic.py:469: FutureWarning: Default multi_class will be changed to 'auto' in
0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[17]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [18]:

```
y_pred = classifier.predict(X_test)
```

In [19]:

```
y_pred
```

Out[19]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 2,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [20]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [21]:

```
cm
```

Out[21]:

```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]], dtype=int64)
```

In [22]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [23]:

```
scores.append(('K Nearest Neighbours', score))
```

In [24]:

```
#Apply support vector machine

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[24]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=0,
    shrinking=True, tol=0.001, verbose=False)
```

In [25]:

```
#Prediction

y_pred = classifier.predict(X_test)
```

In [26]:

```
y_pred
```

Out[26]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [27]:

```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [28]:

```
cm
```

Out[28]:

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

In [29]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [30]:

```
scores.append(('Support Vector Machine', score))
```

In [31]:

```
# Apply Kernel SVM

from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

```
C:\Users\Priyanshi\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: Fu
tureWarning: The default value of gamma will change from 'auto' to 'scale'
in version 0.22 to account better for unscaled features. Set gamma explici
tly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[31]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=0,
    shrinking=True, tol=0.001, verbose=False)
```

In [32]:

```
# Prediction

y_pred = classifier.predict(X_test)
```

In [33]:

```
y_pred
```

Out[33]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [34]:

```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [35]:

```
cm
```

Out[35]:

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

In [36]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [37]:

```
scores.append(('Kernel SVM', score))
```

In [38]:

```
# Apply Naive Bayes

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

Out[38]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [39]:

```
# Prediction

y_pred = classifier.predict(X_test)
```

In [40]:

```
y_pred
```

Out[40]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [41]:

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [42]:

```
cm
```

Out[42]:

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  1,  5]], dtype=int64)
```

In [43]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [44]:

```
scores.append(('Naive bayes', score))
```

In [45]:

```
# Apply Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[45]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=N
one,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [46]:

```
#Prediction

y_pred = classifier.predict(X_test)
```

In [47]:

```
y_pred
```

Out[47]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [49]:

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [50]:

```
cm
```

Out[50]:

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

In [51]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [53]:

```
scores.append(('Decision Tree Classifier', score))
```

In [54]:

```
# Random Forest Classification
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_st
ate = 0)
classifier.fit(X_train, y_train)
```

Out[54]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entro
py',
                       max_depth=None, max_features='auto', max_leaf_nodes
=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbo
se=0,
                       warm_start=False)
```

In [55]:

```
#Prediction

y_pred = classifier.predict(X_test)
```

In [56]:

```
y_pred
```

Out[56]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [57]:

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

In [58]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

In [59]:

```
scores.append(('Random Forest Classifier', score))
```

In [61]:

```
scores
```

Out[61]:

```
[('Logistic Regression', 0.9666666666666667),
 ('K Nearest Neighbours', 0.9666666666666667),
 ('Support Vector Machine', 1.0),
 ('Kernel SVM', 1.0),
 ('Naive bayes', 0.9666666666666667),
 ('Decision Tree Classifier', 1.0),
 ('Random Forest Classifier', 0.9666666666666667)]
```