In [1]:
```
import os
os.chdir('E:\pythonprogs\pythonrepo\Predictive Model')
```

In [5]:
```
os.getcwd()
```

Out[5]:
```
'E:\\pythonprogs\\pythonrepo\\Predictive Model'
```

In [6]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
dataset = pd.read_csv('iris.csv')
```

In [8]:
```
dataset
```

Out[8]:

| | sepal.length | sepal.width | petal.length | petal.width | variety |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Virginica |

150 rows × 5 columns

In [13]:
```
x = dataset.iloc[:,0:4].values
```

In [16]:
```
y = dataset.iloc[:,4]
```

In [18]:
```
y
```

Out[18]:
```
0         Setosa
1         Setosa
2         Setosa
3         Setosa
4         Setosa
         ...
145    Virginica
146    Virginica
147    Virginica
148    Virginica
149    Virginica
Name: variety, Length: 150, dtype: object
```

In [19]:
```
from sklearn.preprocessing import LabelEncoder
```

In [23]:
```
labelencoder_y = LabelEncoder()
y= labelencoder_y.fit_transform(y)
```

In [24]:
```
y
```

Out[24]:
```
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, 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, 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])
```

In [25]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

In [29]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(x_train,y_train)
```

Out[29]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [30]:
```
y_pred = logmodel.predict(x_test)
```

In [36]:
```
y_pred
y_test
```

Out[36]:
```
array([0, 0, 2, 1, 0, 1, 1, 0, 1, 1, 2, 2, 0, 2, 0, 1, 1, 1, 2, 0, 0, 1,
       0, 1, 2, 0, 0, 2, 2, 2])
```

In [32]:
```
from sklearn.metrics import confusion_matrix
```

In [34]:
```
confusion_matrix(y_test,y_pred)
```

Out[34]:
```
array([[11,  0,  0],
       [ 0,  9,  1],
       [ 0,  1,  8]], dtype=int64)
```

In [37]:
```
28/30
```

Out[37]:
```
0.9333333333333333
```

In [42]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier_knn = KNeighborsClassifier(n_neighbors=5, metric='minkowski',p=2)
classifier_knn.fit(x_train,y_train)
```

Out[42]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [44]:
```
y_pred = classifier_knn.predict(x_test)
```

In [45]:
```
confusion_matrix(y_test,y_pred)
```

Out[45]:
```
array([[11,  0,  0],
       [ 0, 10,  0],
       [ 0,  1,  8]], dtype=int64)
```

In [46]:
```
29/30
```

Out[46]:
```
0.9666666666666667
```

In [49]:
```
from sklearn.naive_bayes import GaussianNB
classifier_nb = GaussianNB()
classifier_nb.fit(x_train,y_train)
```

Out[49]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [51]:
```
y_pred = classifier_nb.predict(x_test)
```

In [53]:
```
confusion_matrix(y_test,y_pred)
```

Out[53]:
```
array([[11,  0,  0],
       [ 0,  9,  1],
       [ 0,  1,  8]], dtype=int64)
```

In [54]:
```
28/30
```
Out[54]:
```
0.9333333333333333
```
In [55]:
```
from sklearn.svm import SVC
classifier_svm_sigmoid = SVC(kernel='sigmoid')
classifier_svm_sigmoid.fit(x_train,y_train)
```
Out[55]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```
In [59]:
```
y_pred = classifier_svm_sigmoid.predict(x_test)
```
In [60]:
```
confusion_matrix(y_test,y_pred)
```
Out[60]:
```
array([[ 0,  0, 11],
       [ 0,  0, 10],
       [ 0,  0,  9]], dtype=int64)
```
In [62]:
```
from sklearn.svm import SVC
classifier_svm_linear = SVC(kernel = 'linear')
classifier_svm_linear.fit(x_train,y_train)
```
Out[62]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```
In [63]:
```
y_pred = classifier_svm_linear.predict(x_test)
```
In [67]:
```
confusion_matrix(y_test,y_pred)
```
Out[67]:
```
array([[11,  0,  0],
       [ 0,  9,  1],
       [ 0,  0,  9]], dtype=int64)
```
In [68]:
```
29/30
```
Out[68]:
```
0.9666666666666667
```
In [69]:
```
from sklearn.svm import SVC
classifier_svm_rbf = SVC(kernel='rbf')
classifier_svm_rbf.fit(x_train,y_train)
```
Out[69]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```
In [70]:
```
y_pred = classifier_svm_rbf.predict(x_test)
```
In [71]:
```
confusion_matrix(y_test,y_pred)
```
Out[71]:
```
array([[11,  0,  0],
       [ 0,  9,  1],
       [ 0,  0,  9]], dtype=int64)
```
In [72]:
```
from sklearn.svm import SVC
classifier_svm_poly = SVC(kernel='poly')
classifier_svm_poly.fit(x_train,y_train)
```
Out[72]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```
In [73]:
```
y_pred = classifier_svm_poly.predict(x_test)
```

```
In [74]: confusion_matrix(y_test,y_pred)

Out[74]: array([[11,  0,  0],
                [ 0,  9,  1],
                [ 0,  0,  9]], dtype=int64)
```

```
In [76]: from sklearn.tree import DecisionTreeClassifier
         classifier_dt = DecisionTreeClassifier(criterion = 'entropy')
         classifier_dt.fit(x_train,y_train)

Out[76]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                                max_depth=None, 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='deprecated',
                                random_state=None, splitter='best')
```

```
In [77]: y_pred = classifier_dt.predict(x_test)
```

```
In [78]: confusion_matrix(y_pred,y_test)

Out[78]: array([[11,  0,  0],
                [ 0,  9,  1],
                [ 0,  1,  8]], dtype=int64)
```

```
In [79]: from sklearn.ensemble import RandomForestClassifier
         classifier_rf = RandomForestClassifier(n_estimators=3,criterion='entropy')
         classifier_rf.fit(x_train,y_train)

Out[79]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='entropy', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=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=3,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

```
In [81]: y_pred = classifier_rf.predict(x_test)
```

```
In [82]: //confusion_matrix(y_test,y_pred)

Out[82]: array([[11,  0,  0],
                [ 0,  9,  1],
                [ 0,  1,  8]], dtype=int64)
```

```
In [83]: 28/30

Out[83]: 0.9333333333333333
```

```
In [ ]:
```