Decision Tree

In [1]:

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

In [2]:

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi',
             'pedigree', 'age', 'label']
pima = pd.read_csv("pima-indians-diabetes.csv", header=None,
                   names=col_names)
pima.head()
```

Out[2]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [3]:

```
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp',
                'pedigree']
X = pima[feature_cols]
y = pima.label
```

In [4]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=1)
```

In [5]:

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

In [6]:

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.696969696969697
```

In [7]:

```
 print("Confusion Matrix: ",
        confusion_matrix(y_test, y_pred))

print ("Accuracy : ",
    accuracy_score(y_test,y_pred)*100)

print("Report : ",
    classification_report(y_test, y_pred))
```

```
Confusion Matrix:  [[116  30]
 [ 40  45]]
Accuracy :  69.6969696969697
Report :                precision    recall  f1-score   support

           0       0.74      0.79      0.77       146
           1       0.60      0.53      0.56        85

    accuracy                           0.70       231
   macro avg       0.67      0.66      0.67       231
weighted avg       0.69      0.70      0.69       231
```

In [8]:

```
clf_entropy = DecisionTreeClassifier(
            criterion = "entropy", random_state = 100,
            max_depth = 3, min_samples_leaf = 5)
```

In [9]:

```
clf_entropy.fit(X_train, y_train)
```

Out[9]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5,
                       random_state=100)
```

In [10]:

```
y_pred_entropy = clf_entropy.predict(X_test)
y_pred_entropy
```

Out[10]:

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

In [ ]: