In [2]:
```
import pandas as pd
import pylab as pl
import numpy as np
import scipy.optimize as opt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
import warnings
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn import tree
import graphviz
import pydotplus
from PIL import Image
# Adicion de colores al arbol de decision
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
warnings.filterwarnings('ignore')
import os
os.chdir('/Users/Lenovo/Desktop/EBAC')
```

In [4]:
```
data = pd.read_csv('drugs1.csv')
data
```

Out[4]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

In [6]:
```
features_cols = ['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']
X = data[features_cols].values
y = data.Drug
```

In [8]:
```
from sklearn import preprocessing
Cod_Sex = preprocessing.LabelEncoder()
Cod_Sex.fit(['F', 'M'])
X[:,1] = Cod_Sex.transform(X[:,1])

Cod_BP = preprocessing.LabelEncoder()
Cod_BP.fit(['HIGH', 'LOW', 'NORMAL'])
X[:,2] = Cod_BP.transform(X[:,2])

Cod_Cholesterol = preprocessing.LabelEncoder()
Cod_Cholesterol.fit(['HIGH', 'LOW', 'NORMAL'])
X[:,3] = Cod_Cholesterol.transform(X[:,3])
```

In [10]:
```
# Creacion de grupos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
```

Bosques Aleatorios (Random Forest)

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 100, random_state = 1)
rf.fit(X_train, y_train)
```

Out[13]:
```
RandomForestClassifier
RandomForestClassifier(random_state=1)
```

In [15]:
```
# Score F1 para el grupo de entrenamiento (training)
rf.score(X_train, y_train)
```

Out[15]: 1.0

In [17]:
```
# Score F1 para el grupo de prueba (testing)
rf.score(X_test, y_test)
```

Out[17]: 0.95

In [19]:
```
# Estadisticas de desmpeño generales
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       drugA       0.67      1.00      0.80         4
       drugB       1.00      0.67      0.80         6
       drugC       1.00      0.75      0.86         4
       drugX       0.95      1.00      0.97        19
       drugY       1.00      1.00      1.00        27

    accuracy                           0.95        60
   macro avg       0.92      0.88      0.89        60
weighted avg       0.96      0.95      0.95        60
```

Gradient Boosted trees

In [22]:
```
from sklearn.ensemble import GradientBoostingClassifier
# Creacion de grupos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
gbt = GradientBoostingClassifier(n_estimators = 100, random_state = 1) # 100 arboles de profundidad de 3 (Defau
gbt.fit(X_train, y_train)
```

Out[22]:
```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=1)
```

In [24]:
```
print('Training F1 Score: ,', gbt.score(X_train, y_train))
print('Testing F1 Score: ,', gbt.score(X_test, y_test))
```

```
Training F1 Score: , 1.0
Testing F1 Score: , 1.0
```

In [26]:
```
# estadisticas de desempe;o general
y_pred = gbt.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       drugA       1.00      1.00      1.00         4
       drugB       1.00      1.00      1.00         2
       drugC       1.00      1.00      1.00         4
       drugX       1.00      1.00      1.00        13
       drugY       1.00      1.00      1.00        17

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

AdaBoost Classifier

In [29]:
```
from sklearn.ensemble import AdaBoostClassifier
# Creacion de grupos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
# AdaBoost usa arboles de decision como clasficador por Default
abc = AdaBoostClassifier(n_estimators = 50, learning_rate = 1)

# Entrenamiento del clasificador AdaBoost
model = abc.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       drugA       0.67      1.00      0.80         4
       drugB       0.00      0.00      0.00         2
       drugC       0.00      0.00      0.00         4
       drugX       0.76      1.00      0.87        13
       drugY       1.00      1.00      1.00        17

    accuracy                           0.85        40
   macro avg       0.49      0.60      0.53        40
weighted avg       0.74      0.85      0.79        40
```

Conclusion

Despues de haber evaluado los 3 modelos vistos anteriormente, llegue a la conclusion que en general los 3 son muy buenos en este cas para la base de informacion con la que estamos trabajando ya que, el accuracy va desde un 85% minimo, hasta el 100%. En este caso me quedaria con Gradient Boosted Trees, ya que pudo acertar en el 100% de los resultados a predecir.

El poder predictivo comparado con Arboles de Decision, si mejoro en este caso con Gradient Boosted Trees, ya que llegamos al 100% y anteriormente el valor mas alto era de 97%, aun asi para este tipo de informacion yo creo que todos estos modelos son muy buenos por su alta acertividad.

In [ ]: