In [1]:
```python
import pandas as pd
import pylab as pl
import numpy as np
from sklearn.linear_model import LogisticRegression
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
warnings.filterwarnings('ignore')
import os
os.chdir('/Users/Lenovo/Desktop/EBAC')
```

In [3]:
```python
data = pd.read_csv('drugs1.csv')
data
```

Out[3]:

	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 [5]:
```python
features_cols = ['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']
X = data[features_cols].values
y = data.Drug
```

In [7]:
```python
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 [9]:
```python
# 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)
```

Regresion logistica con Solucionador (Solver): sag

In [12]:
```python
model = LogisticRegression(solver = 'sag')
clf = model.fit(X_train, y_train)

# Prediccion de etiquetas de clase sobre datos de prueba
y_pred = model.predict(X_test)

# Impresion de coeficientes de la regresion de puntajes
print('Intercepto (Beta0): ', clf.intercept_)
print('Pesos de cada variable (Beta1, Beta2, ..., Beta7): ', clf.coef_)

print('-----------------------------------------------------------------------------------------------------------

# Evaluacion de la precision del modelo
score = model.score(X_test, y_test)
print('Score de precision: ', score)
```

```
print('-------------------------------------------------------------------------------------------------------

# Reporte de clasificacion
print(classification_report(y_test, y_pred))
```

```
Intercepto (Beta0):  [ 0.25207028 -0.08295724  0.09391024  0.21916712 -0.48219041]
Pesos de cada variable (Beta1, Beta2, ..., Beta7):  [[-0.02543513  0.15505068 -0.63735024  0.12174788  0.1007136
3]
 [ 0.07208856 -0.06031722 -0.51824822 -0.11326672 -0.29048744]
 [-0.00437994  0.01393438  0.14886401 -0.39560134  0.00341319]
 [ 0.01153286 -0.08992082  1.17809626  0.53268686 -0.12734421]
 [-0.05380635 -0.01874702 -0.17136182 -0.14556668  0.31370484]]
----------------------------------------------------------------------------------------------------------
------------------------------
Score de precision:  0.7833333333333333
----------------------------------------------------------------------------------------------------------
------------------------------
              precision    recall  f1-score   support

       drugA       0.33      0.25      0.29         4
       drugB       0.57      0.67      0.62         6
       drugC       0.00      0.00      0.00         4
       drugX       0.85      0.89      0.87        19
       drugY       0.83      0.93      0.88        27

    accuracy                           0.78        60
   macro avg       0.52      0.55      0.53        60
weighted avg       0.72      0.78      0.75        60
```

Resgresion logistica con Solucionador (Solver): newton-cg

In [26]:

```
model = LogisticRegression(solver = 'newton-cg')
clf = model.fit(X_train, y_train)

# Prediccion de etiquetas de clase sobre datos de prueba
y_pred = model.predict(X_test)

# Impresion de coeficientes de la regresion de puntajes
print('Intercepto (Beta0): ', clf.intercept_)
print('Pesos de cada variable (Beta1, Beta2, ..., Beta7): ', clf.coef_)

print('-------------------------------------------------------------------------------------------------------

# Evaluacion de la precision del modelo
score = model.score(X_test, y_test)
print('Score de precision: ', score)

print('-------------------------------------------------------------------------------------------------------

# Reporte de clasificacion
print(classification_report(y_test, y_pred))
```

```
Intercepto (Beta0):  [ 14.48790715  -0.08512375  10.50580619   5.75638881 -30.6649784 ]
Pesos de cada variable (Beta1, Beta2, ..., Beta7):  [[-8.96856681e-02  6.77268617e-02 -2.33960214e+00 -2.3674471
8e-01
  -6.33653119e-01]
 [ 1.42467196e-01 -1.52212913e-01 -1.59788400e+00 -1.32018905e-01
  -4.42440147e-01]
 [-3.38662547e-02  2.54380976e-04  4.21069578e-01 -1.14094176e+00
  -5.85695514e-01]
 [-7.31005987e-03 -3.94719449e-01  3.09381357e+00  1.26917979e+00
  -5.67169846e-01]
 [-1.16052125e-02  4.78951120e-01  4.22602991e-01  2.40525596e-01
   2.22895863e+00]]
----------------------------------------------------------------------------------------------------------
------------------------------
Score de precision:  0.95
----------------------------------------------------------------------------------------------------------
------------------------------
              precision    recall  f1-score   support

       drugA       0.80      1.00      0.89         4
       drugB       0.71      0.83      0.77         6
       drugC       1.00      0.50      0.67         4
       drugX       1.00      1.00      1.00        19
       drugY       1.00      1.00      1.00        27

    accuracy                           0.95        60
   macro avg       0.90      0.87      0.86        60
weighted avg       0.96      0.95      0.95        60
```

Resgresion logistica con Solucionador (Solver): liblinear

In [18]:
```
model = LogisticRegression(solver = 'liblinear')
clf = model.fit(X_train, y_train)

# Prediccion de etiquetas de clase sobre datos de prueba
y_pred = model.predict(X_test)

# Impresion de coeficientes de la regresion de puntajes
print('Intercepto (Beta0): ', clf.intercept_)
print('Pesos de cada variable (Beta1, Beta2, ..., Beta7): ', clf.coef_)

print('----------------------------------------------------------------------------------------------------------

# Evaluacion de la precision del modelo
score = model.score(X_test, y_test)
print('Score de precision: ', score)

print('----------------------------------------------------------------------------------------------------------

# Reporte de clasificacion
print(classification_report(y_test, y_pred))
```

```
Intercepto (Beta0):  [ 2.31089578 -0.65080868  0.6825165   0.16513608 -4.27580351]
Pesos de cada variable (Beta1, Beta2, ..., Beta7):  [[-0.02428697  0.42805745 -2.29889241 -0.14361067 -0.1657302
8]
 [ 0.11187746 -0.52110997 -2.33619921 -0.54425809 -0.45569985]
 [-0.00360594  0.0611196   0.01560035 -1.39441786 -0.17622994]
 [ 0.01055564 -0.71011324  3.12578883  1.45357325 -0.47400147]
 [-0.0414012   0.16666879 -0.24890702 -0.21347592  0.42761027]]
------------------------------------------------------------------------------------------------------------------
------------------------------
Score de precision:  0.85
------------------------------------------------------------------------------------------------------------------
------------------------------
              precision    recall  f1-score   support

       drugA       0.60      0.75      0.67         4
       drugB       0.57      0.67      0.62         6
       drugC       1.00      0.25      0.40         4
       drugX       0.86      1.00      0.93        19
       drugY       0.96      0.89      0.92        27

    accuracy                           0.85        60
   macro avg       0.80      0.71      0.71        60
weighted avg       0.87      0.85      0.84        60
```

Resgresion logistica con Solucionador (Solver): saga

In [21]:
```
model = LogisticRegression(solver = 'saga')
clf = model.fit(X_train, y_train)

# Prediccion de etiquetas de clase sobre datos de prueba
y_pred = model.predict(X_test)

# Impresion de coeficientes de la regresion de puntajes
print('Intercepto (Beta0): ', clf.intercept_)
print('Pesos de cada variable (Beta1, Beta2, ..., Beta7): ', clf.coef_)

print('----------------------------------------------------------------------------------------------------------

# Evaluacion de la precision del modelo
score = model.score(X_test, y_test)
print('Score de precision: ', score)

print('----------------------------------------------------------------------------------------------------------

# Reporte de clasificacion
print(classification_report(y_test, y_pred))
```

```
Intercepto (Beta0):  [ 0.1265274  -0.0522943   0.04611565  0.13371509 -0.25406383]
Pesos de cada variable (Beta1, Beta2, ..., Beta7):  [[-0.02337324  0.08785222 -0.37363311  0.08269469  0.0892573
2]
 [ 0.06354411 -0.03265927 -0.3150302  -0.06978496 -0.27327782]
 [-0.00356606  0.00812699  0.07605    -0.24470664 -0.00516823]
 [ 0.01628445 -0.04410388  0.74523258  0.35058191 -0.09611223]
 [-0.05288925 -0.01921606 -0.13261927 -0.118785    0.28530096]]
----------------------------------------------------------------------------------------------------------
------------------------------
Score de precision:  0.75
----------------------------------------------------------------------------------------------------------
------------------------------
              precision    recall  f1-score   support

       drugA       0.00      0.00      0.00         4
       drugB       0.57      0.67      0.62         6
       drugC       0.00      0.00      0.00         4
       drugX       0.75      0.79      0.77        19
       drugY       0.79      0.96      0.87        27

    accuracy                           0.75        60
   macro avg       0.42      0.48      0.45        60
weighted avg       0.65      0.75      0.70        60
```

Resgresion logistica con Solucionador (Solver): lbfgs

In [24]:

```python
model = LogisticRegression(solver = 'lbfgs')
clf = model.fit(X_train, y_train)

# Prediccion de etiquetas de clase sobre datos de prueba
y_pred = model.predict(X_test)

# Impresion de coeficientes de la regresion de puntajes
print('Intercepto (Beta0): ', clf.intercept_)
print('Pesos de cada variable (Beta1, Beta2, ..., Beta7): ', clf.coef_)

print('-----------------------------------------------------------------------------------------------------------

# Evaluacion de la precision del modelo
score = model.score(X_test, y_test)
print('Score de precision: ', score)

print('-----------------------------------------------------------------------------------------------------------

# Reporte de clasificacion
print(classification_report(y_test, y_pred))
```

```
Intercepto (Beta0):  [ 1.16939214 -0.31971945  0.54355859  0.7087925  -2.10202379]
Pesos de cada variable (Beta1, Beta2, ..., Beta7):  [[-0.03833878  0.47809653 -2.01089716 -0.08237991  0.1626778
7]
 [ 0.09852575 -0.23784054 -1.80071268 -0.39446806 -0.28193157]
 [ 0.0100157   0.0773516   0.49128879 -1.13697752 -0.04849119]
 [-0.01344921 -0.43843358  3.43060162  1.59338483 -0.30371246]
 [-0.05675346  0.12082599 -0.11028058  0.02044065  0.47145735]]
----------------------------------------------------------------------------------------------------------
------------------------------
Score de precision:  0.8333333333333334
----------------------------------------------------------------------------------------------------------
------------------------------
              precision    recall  f1-score   support

       drugA       0.60      0.75      0.67         4
       drugB       0.57      0.67      0.62         6
       drugC       0.00      0.00      0.00         4
       drugX       0.83      1.00      0.90        19
       drugY       0.96      0.89      0.92        27

    accuracy                           0.83        60
   macro avg       0.59      0.66      0.62        60
weighted avg       0.79      0.83      0.81        60
```

Conclusion

Despues de haber evaluado todos los modelos, el mejor modelo de acuerdo al accuracy en F1 Score, fue el de Newton-cg, ya que alcanzo un 95% de acertividad en su pronostico.

Eficacia del modelo Newton-cg

In [44]:
```
classes = np.unique(y_test)
n_classes = len(classes)
```

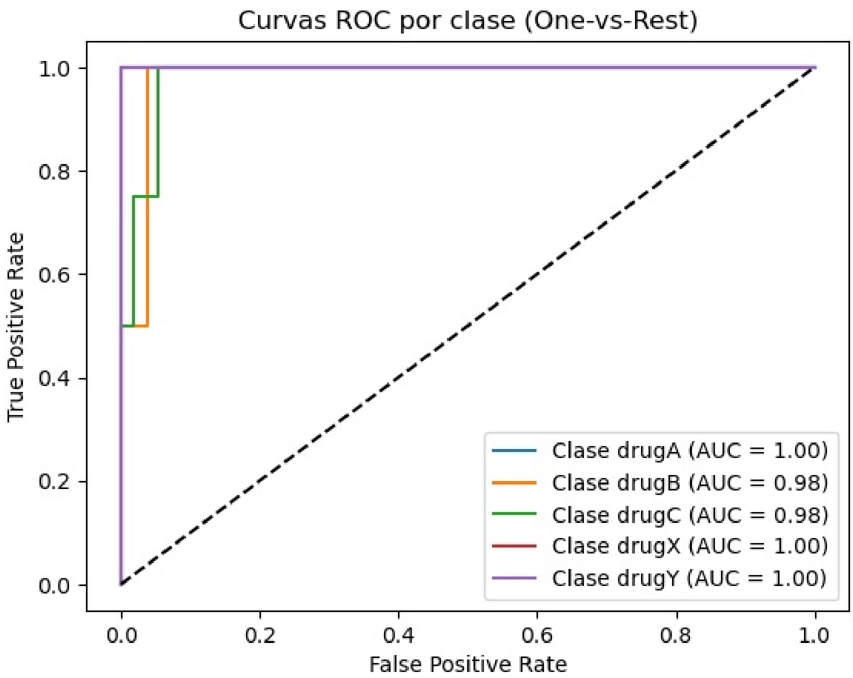
In [48]:
```
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc, roc_auc_score

y_test_bin = label_binarize(y_test, classes=classes)
```

In [50]:
```
y_score = model.predict_proba(X_test)
```

In [58]:
```
for i in range(n_classes):
    fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'Clase {classes[i]} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Curvas ROC por clase (One-vs-Rest)')
plt.legend(loc='lower right')
plt.show()
```



En la grafica ROC podemos observar que cada una de las clases esta muy pegadas a los bordes, por lo que podemos confirmar que la eficiencia en el modelo es muy alta.

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js