Anexo IV

September 25, 2023

1 Metodos de clasificación

1.1 Importacion de libraries

[1]:
```
import numpy as np
import pandas as pd
```

1.2 Tratamiento de los datos previo a implementar los modelos

1.2.1 Paso 1) Importamos el fichero csv.

[2]:
```
ds_stroke = pd.read_csv("/home/guincho/Desktop/stroke_final.csv")
ds_stroke
```

[2]:

	age	hypertension	heart_disease	ever_married	work_type	\
0	67.0	0	1	1	Private	
1	61.0	0	0	1	Self-employed	
2	80.0	0	1	1	Private	
3	49.0	0	0	1	Private	
4	79.0	1	0	1	Self-employed	
…	…	…	…	…	…	
5105	80.0	1	0	1	Private	
5106	81.0	0	0	1	Self-employed	
5107	35.0	0	0	1	Self-employed	
5108	51.0	0	0	1	Private	
5109	44.0	0	0	1	Govt_job	

	avg_glucose_level	bmi	smoking_status	stroke
0	228.69	36.6	formerly smoked	1
1	202.21	28.1	never smoked	1
2	105.92	32.5	never smoked	1
3	171.23	34.4	smokes	1
4	174.12	24.0	never smoked	1
…	…	…	…	…
5105	83.75	28.1	never smoked	0
5106	125.20	40.0	never smoked	0
5107	82.99	30.6	never smoked	0
5108	166.29	25.6	formerly smoked	0

```
5109              85.28  26.2          Unknown       0

[5110 rows x 9 columns]
```

1.2.2 Paso 2) Separar la variable a predecir (y = stroke) de las predictoras

stroke es la ultima columna del dataset. Estamos creando dos matrices

[3]:
```
x = ds_stroke.iloc[:, 0:-1].values
y = ds_stroke.iloc[:, -1].values
```

[4]:
```
x
```

[4]:
```
array([[67.0, 0, 1, …, 228.69, 36.6, 'formerly smoked'],
       [61.0, 0, 0, …, 202.21, 28.1, 'never smoked'],
       [80.0, 0, 1, …, 105.92, 32.5, 'never smoked'],
       …,
       [35.0, 0, 0, …, 82.99, 30.6, 'never smoked'],
       [51.0, 0, 0, …, 166.29, 25.6, 'formerly smoked'],
       [44.0, 0, 0, …, 85.28, 26.2, 'Unknown']], dtype=object)
```

[5]:
```
x[0]
```

[5]:
```
array([67.0, 0, 1, 1, 'Private', 228.69, 36.6, 'formerly smoked'],
      dtype=object)
```

1.2.3 Paso 3) Tranformar los datos ctegóricos con OneHotEncoder

Nuestros datos categóricos son work_type & smoking_status Son las columnas 4 & 7

[6]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

columna = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),␣
 ↪[4,7])], remainder = 'passthrough')

x = np.array(columna.fit_transform(x))
```

[7]:
```
x[0]
```

[7]:
```
array([0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 67.0, 0, 1, 1, 228.69,
       36.6], dtype=object)
```

[8]:
```
print('Forma de X: ', x.shape)
print('Forma de Y: ', y.shape)
```

```
Forma de X:  (5110, 15)
Forma de Y:  (5110,)
```

1.2.4 Paso 4) Escalado de variables

[9]:
```
from sklearn.preprocessing import StandardScaler

es = StandardScaler()

x = es.fit_transform(x)
```

1.2.5 Paso 5) Dividimos entre Training y Test

[10]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
 ↪random_state = 123)
```

[11]:
```
print('Predictoras train {}'.format(x_train.shape))
print('Predictoras test {}'.format(x_test.shape))
print('Target train {}'.format(y_train.shape))
print('Target test {}'.format(y_test.shape))
```

```
Predictoras train (3577, 15)
Predictoras test (1533, 15)
Target train (3577,)
Target test (1533,)
```

1.2.6 Paso 6) Balanceando los datos con SMOTE

Las observaciones con Stroke = 1 son mucho menores que aquellas con Stroke = 0.

[12]:
```
from imblearn.over_sampling import SMOTE
```

[17]:
```
print("Accidentes cardiovasculares == '1', antes del SMOTE: {}".
 ↪format(sum(y_train==1)))
print("Accidentes cardiovasculares == '0', antes del SMOTE: {}\n".
 ↪format(sum(y_train==0)))

smote = SMOTE(random_state = 123)
x_train_s, y_train_s = smote.fit_resample(x_train, y_train.ravel())

print("Accidentes cardiovasculares == '1', despues del SMOTE: {}".
 ↪format(sum(y_train_res==1)))
print("Accidentes cardiovasculares == '0', despues del SMOTE: {}\n".
 ↪format(sum(y_train_res==0)))
```

```
Accidentes cardiovasculares == '1', antes del SMOTE: 181
Accidentes cardiovasculares == '0', antes del SMOTE: 3396

Accidentes cardiovasculares == '1', despues del SMOTE: 3396
```

```
Accidentes cardiovasculares == '0', despues del SMOTE: 3396
```

1.3 Modelos de clasificación

Vamos a usar 3 modelos de clasificación. Serán:

1) RandomForest. Para ver si nos da mejor que en RStudio
2) BernoulliNB. Algoritmo de clasificacíon basado en el teorema de bayes. Empleado mucho en características binarias
3) SVC (Support vector classifier). El objetivo de un SVC es encontrar el hiperplano que mejor separa las clases en un espacio dimensional

[18]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.svm import SVC
```

[20]:
```
from sklearn.metrics import recall_score, precision_score, confusion_matrix,␣
 ↪classification_report
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score,␣
 ↪roc_auc_score, roc_curve

from sklearn.model_selection import cross_val_score
```

[39]:
```
modelos = []
modelos.append(["Bosque aleatorio", RandomForestClassifier(random_state = 123)])
modelos.append(["SVM", SVC(random_state=123)])
modelos.append(["BernoulliNB", BernoulliNB()])


for i in range(len(modelos)):
    m = modelos[i][1]
    m.fit(x_train_res, y_train_res)
    y_pred = m.predict(x_test)
    matriz_conf = confusion_matrix(y_test, y_pred)  # Matriz de confusion
    roc = roc_auc_score(y_test, y_pred)  #ROC AUC Score

    # Calculando Sensibilidad y Especificidad
    tn, fp, fn, tp = matriz_conf.ravel()
    sensibilidad = tp / (tp + fn) * 100
    especificidad = tn / (tn + fp) * 100

    print(modelos[i][0],':')
    print('Matriz de Confusión:')
    print(matriz_conf)
    print('Valor de Accuracy: {:.3f}'.format(accuracy_score(y_test, y_pred)))
    print('Valor de ROC AUC {:.3f}'.format(roc))
    print('Sensibilidad: {:.2f}%'.format(sensibilidad))
```

```
    print('Especificidad: {:.2f}%'.format(especificidad))
    print('\n')
```

```
Bosque aleatorio :
Matriz de Confusión:
[[1350  115]
 [  50   18]]
Valor de Accuracy: 0.892
Valor de ROC AUC 0.593
Sensibilidad: 26.47%
Especificidad: 92.15%


SVM :
Matriz de Confusión:
[[1166  299]
 [  32   36]]
Valor de Accuracy: 0.784
Valor de ROC AUC 0.663
Sensibilidad: 52.94%
Especificidad: 79.59%


BernoulliNB :
Matriz de Confusión:
[[799 666]
 [  9  59]]
Valor de Accuracy: 0.560
Valor de ROC AUC 0.707
Sensibilidad: 86.76%
Especificidad: 54.54%
```

1.4 Conclusiones

Si consideramos principalmente el AUC-ROC como métrica de rendimiento, el modelo de Bernoulli Naive Bayes parece ser la mejor opción en este caso

El modelo Bernoulli Naive Bayes muestra una alta sensibilidad (86.76%), lo que indica que es bueno identificando verdaderos positivos. Sin embargo, su especificidad es relativamente baja (54.54%)

El modelo SVM muestra una sensibilidad razonable (52.94%) y una especificidad aceptable (79.59%).

Random forest no es un buen modelo ya que su accuracy es muy cercana a 0.5 y por tanto al azar

[ ]: