Regresión Logística

Juliho Castillo Colmenares

Instrucciones

¡Bienvenidos a la actividad práctica del módulo!

Antes de empezar

El objetivo de esta actividad es que efectúes un análisis de clasificación mediante Árboles de decisión que permita desarrollar un modelo predictivo basado en distintas métricas aplicadas a una base de datos grande.

Entregable: Un Jupyter Notebook (archivo de extensión .ipynb), archivo PDF y capturas de pantalla en el espacio de respuesta que muestren tanto el código desarrollado como la solución al problema planteado, incluyendo los comentarios que sean pertinentes a las preguntas que se plantean. Entregas sin estos elementos no serán calificadas.

¡Buena suerte!

Problema: Imagina que eres un investigador médico que recopila datos para un estudio. Has recopilado datos sobre un conjunto de pacientes, todos ellos con la misma enfermedad. Durante su tratamiento, cada paciente ha respondido a uno de los 5 medicamentos, el fármaco A, el fármaco B, el fármaco c, el fármaco X y el Y.

Parte de tu trabajo consiste en construir un modelo para averiguar qué medicamento podría ser apropiado para un futuro paciente con la misma enfermedad. Los conjuntos de características de este conjunto de datos son la edad, el sexo, la presión arterial y el colesterol de los pacientes, y el objetivo es estudiar el fármaco al que respondió cada paciente.

Se trata de un ejemplo de clasificador binario, y se puede utilizar la parte de entrenamiento del conjunto de datos para construir un árbol de decisión, y luego utilizarlo para predecir la clase de un paciente desconocido, o para prescribirlo a un nuevo paciente.

Variable a pronosticar:

Age	Sex	BP	Cholesterol	Na_to_K	Drug
23	F	HIGH	HIGH	25.355	drugY
47	M	LOW	HIGH	13.093	drugC
47	M	LOW	HIGH	10.114	drugC
28	F	NORMAL	HIGH	7.798	drugX
61	F	LOW	HIGH	18.043	drugY
22	F	NORMAL	HIGH	8.607	drugX
49	F	NORMAL	HIGH	16.275	drugY
41	M	LOW	HIGH	11.037	drugC

Paso 1

Cargue la base de datos "drugs.csv" en Python e investigue cómo convertir las variables predictoras cualitativas de esta base a una escala numérica mediante la instrucción "preprocessing.LabelEncoder()". Por ejemplo, si una variable tiene 3 posibles categorías, deberá cambiar sus resultados a 0, 1 o 2.

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("drugs.csv")
df.head()
```

Out[ ]:

	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

In [ ]:
```
X = df.drop("Drug", axis=1)
y = df["Drug"]
```

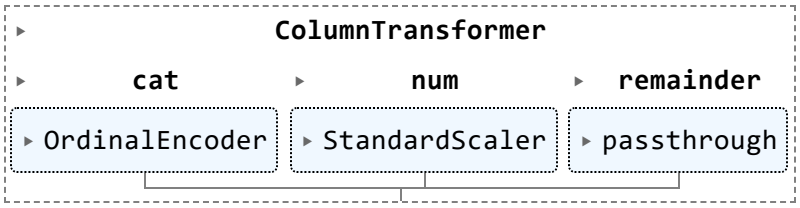
In [ ]:
```
from preprocess import create_preprocessing_pipeline
```

```
preprocess_pipeline = create_preprocessing_pipeline(
    categorical_variables=["Sex", "BP", "Cholesterol"],
    numerical_variables=["Age", "Na_to_K"]
    )
```

In [ ]:
```
X_preprocessed = preprocess_pipeline.fit_transform(X)
```

In [ ]:
```
preprocessor = preprocess_pipeline.named_steps['preprocessor']
preprocessor
```

Out[ ]:



In [ ]:
```
feature_names = preprocessor.get_feature_names_out()
feature_names
```

Out[ ]:
```
array(['cat__Sex', 'cat__BP', 'cat__Cholesterol', 'num__Age',
       'num__Na_to_K'], dtype=object)
```

In [ ]:
```
ordinal_encoder = preprocessor.transformers_[0][1]
categories = ordinal_encoder.categories_
categories
```

Out[ ]:
```
[array(['F', 'M'], dtype=object),
 array(['HIGH', 'LOW', 'NORMAL'], dtype=object),
 array(['HIGH', 'NORMAL'], dtype=object)]
```

Paso 2

Use los diversos métodos de optimización para una Regresión Logística vistos en este módulo para encontrar un algoritmo óptimo de clasificación. Explique cuál sería su recomendación para este caso. Regresión Logística

In [ ]:
```
# from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings("ignore")


# Definir el espacio de búsqueda de parámetros
param_grid = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 200, 300]
}

# Configurar GridSearchCV
grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5, scoring='accuracy')

# Ajustar GridSearchCV
```

```
grid_search.fit(X_preprocessed, y)

# Mejores parámetros y puntuación
print("Mejores parámetros:", grid_search.best_params_)
print("Mejor puntuación:", grid_search.best_score_)
```

```
Mejores parámetros: {'C': 100, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear'}
Mejor puntuación: 0.9800000000000001
```

In [ ]:
```
model = LogisticRegression(**grid_search.best_params_)
model.fit(X_preprocessed, y)
```

Out[ ]:
```
LogisticRegression(C=100, penalty='l1', solver='liblinear')
```

Paso 3

¿Qué tan eficaz es el algoritmo predictivo escogido? Explique a detalle comentando sobre los indicadores obtenidos mediante el reporte de clasificación correspondiente y la curva ROC.

Vamos a crear conjuntos de entrenamiento y de prueba para evitar el sobreajuste obtenido al entrenar el modelo con el conjunto de datos completo.

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


(
    X_train,
    X_test,
    y_train,
    y_test
) = train_test_split(X_preprocessed, y, test_size=0.2, random_state=42)
```

In [ ]:
```
model.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression(C=100, penalty='l1', solver='liblinear')
```

In [ ]:
```
from sklearn.metrics import classification_report, roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Predicciones del modelo
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)
```

In [ ]:
```
# Reporte de clasificación
print("Reporte de Clasificación:")
print(classification_report(y_test, y_pred))
```

```
Reporte de Clasificación:
              precision    recall  f1-score   support

       drugA       1.00      0.83      0.91         6
       drugB       0.75      1.00      0.86         3
       drugC       1.00      0.20      0.33         5
       drugX       0.73      1.00      0.85        11
       drugY       1.00      1.00      1.00        15

    accuracy                           0.88        40
   macro avg       0.90      0.81      0.79        40
weighted avg       0.91      0.88      0.85        40
```

El reporte de clasificación proporciona varias métricas clave para evaluar el rendimiento de un modelo de clasificación. Aquí está la interpretación de cada métrica:

1. **Precision (Precisión)**: La precisión es la proporción de verdaderos positivos sobre el total de predicciones positivas. Indica cuántas de las predicciones positivas del modelo son realmente correctas.
    - Ejemplo: Para `drugA`, la precisión es 1.00, lo que significa que todas las predicciones positivas para `drugA` fueron correctas.
2. **Recall (Sensibilidad o Recall)**: El recall es la proporción de verdaderos positivos sobre el total de verdaderos positivos y falsos negativos. Indica cuántos de los casos positivos reales fueron capturados por el modelo.
    - Ejemplo: Para `drugB`, el recall es 1.00, lo que significa que el modelo identificó correctamente todos los casos de `drugB`.
3. **F1-score**: El F1-score es la media armónica de la precisión y el recall. Proporciona un balance entre precisión y recall, especialmente útil cuando hay una distribución desigual de clases.
    - Ejemplo: Para `drugC`, el F1-score es 0.33, lo que indica un bajo rendimiento debido a un recall muy bajo (0.20).
4. **Support (Soporte)**: El soporte es el número de ocurrencias reales de la clase en el conjunto de datos.
    - Ejemplo: Para `drugY`, el soporte es 15, lo que significa que hay 15 instancias de `drugY` en el conjunto de datos.
5. **Accuracy (Exactitud)**: La exactitud es la proporción de todas las predicciones correctas (tanto verdaderos positivos como verdaderos negativos) sobre el total de instancias.
    - En este caso, la exactitud es 0.88, lo que significa que el modelo clasificó correctamente el 88% de las instancias.
6. **Macro avg (Promedio macro)**: El promedio macro calcula la media de las métricas (precisión, recall, F1-score) para cada clase sin tener en cuenta el soporte. Es útil para evaluar el rendimiento del modelo en clases desbalanceadas.
    - Ejemplo: El promedio macro de precisión es 0.90, lo que indica un buen rendimiento general en términos de precisión.

7. **Weighted avg (Promedio ponderado)**: El promedio ponderado calcula la media de las métricas (precisión, recall, F1-score) ponderadas por el soporte de cada clase. Es útil para evaluar el rendimiento del modelo teniendo en cuenta la distribución de clases.

    - Ejemplo: El promedio ponderado de F1-score es 0.85, lo que indica un buen rendimiento general del modelo.

En resumen, el modelo tiene un buen rendimiento general con una exactitud del 88%, pero hay variaciones significativas en el rendimiento entre las diferentes clases, especialmente para `drugC`, que tiene un bajo recall y F1-score.

In [ ]:
```
# Curva ROC
fpr, tpr, _ = roc_curve(y_test, y_pred_proba[:, 1], pos_label=model.classes_[1])
```

In [ ]:
```
print(f"fpr: {fpr}")
```

```
fpr: [0.         0.         0.         0.02702703 0.02702703 1.        ]
```

El resultado `fpr` (False Positive Rate) es una lista de valores que representan la tasa de falsos positivos en diferentes umbrales de decisión para un modelo de clasificación. Aquí está la interpretación de los valores:

- `0.0`: En los primeros tres umbrales, la tasa de falsos positivos es 0, lo que significa que no hay falsos positivos.
- `0.02702703`: En el cuarto y quinto umbral, la tasa de falsos positivos es aproximadamente 2.7%, lo que indica que el 2.7% de las muestras negativas fueron clasificadas incorrectamente como positivas.
- `1.0`: En el último umbral, la tasa de falsos positivos es 100%, lo que significa que todas las muestras negativas fueron clasificadas incorrectamente como positivas.

In [ ]:
```
print(f"tpr: {tpr}")
```

```
tpr: [0.         0.33333333 0.66666667 0.66666667 1.         1.        ]
```

El resultado `tpr` (True Positive Rate) es una lista de valores que representan la tasa de verdaderos positivos en diferentes umbrales de decisión para un modelo de clasificación. Aquí está la interpretación de los valores:

- `0.0`: En el primer umbral, la tasa de verdaderos positivos es 0, lo que significa que no hay verdaderos positivos.
- `0.33333333`: En el segundo umbral, la tasa de verdaderos positivos es aproximadamente 33.3%, lo que indica que el 33.3% de las muestras positivas fueron clasificadas correctamente.
- `0.66666667`: En el tercer umbral, la tasa de verdaderos positivos es aproximadamente 66.7%, lo que indica que el 66.7% de las muestras positivas fueron clasificadas correctamente.
- `0.66666667`: En el cuarto umbral, la tasa de verdaderos positivos se mantiene en aproximadamente 66.7%.

- `1.0` : En el quinto y sexto umbral, la tasa de verdaderos positivos es 100%, lo que significa que todas las muestras positivas fueron clasificadas correctamente.

In [ ]:
```
roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovo')
print(f"ROC AUC Score: {roc_auc}")
```

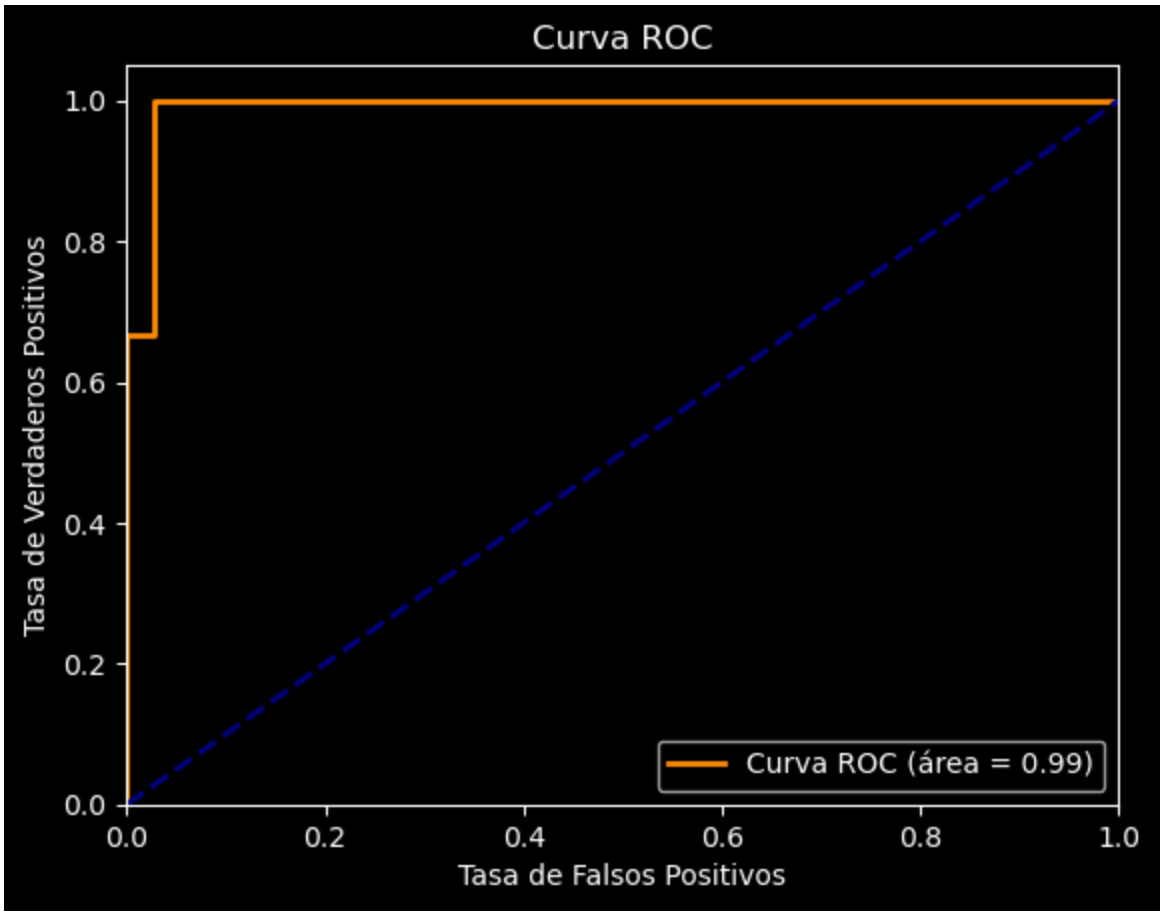
```
ROC AUC Score: 0.9949999999999999
```

El resultado `ROC AUC Score: 0.9949999999999999` indica que el área bajo la curva ROC (Receiver Operating Characteristic) es aproximadamente 0.995. Aquí está la interpretación:

- **ROC AUC Score**: Es una métrica que mide el rendimiento de un modelo de clasificación. El valor varía entre 0 y 1.
- **0.995**: Un valor cercano a 1 indica un excelente rendimiento del modelo. Significa que el modelo tiene una alta capacidad para distinguir entre las clases positivas y negativas.

En resumen, un ROC AUC Score de 0.995 sugiere que el modelo de regresión logística tiene un rendimiento muy alto en términos de clasificación.

In [ ]:
```
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='Curva ROC (área = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()
```


Curva ROC
Tasa de Verdaderos Positivos
Tasa de Falsos Positivos
0.0
0.2
0.4
0.6
0.8
1.0
Curva ROC (área = 0.99)