Regresión Logística

Tinoco Sergio, Sánchez Flavio

Instituto Politécnico Nacional, Escuela Superior de Cómputo

Aprendizaje de Máquina 5BV1

13 de enero de 2023

I. INTRODUCCIÓN

Uno de los modelos de clasificación en el aprendizaje de maquina es la regresión logística, el cual es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categoría binaria en función de las variables independientes o predictoras.

Este tipo de regresión se caracteriza por hacer uso de una función matemática llamada función “Sigmoide” la cual adquiere valores que van desde 0 hasta 1 comportándose como una ‘S’ en todo su dominio.

De igual forma este tipo de modelos de clasificación se aplica para problemas “lineales”, lo que implica que el conjunto de datos debe ser linealmente separable.

En esta práctica se implementará el algoritmo de regresión logística en los lenguajes de programación Python y R.

II. OBJETIVO

Comprender la implementación de la regresión logística en R y Python.

III. MARCO TEÓRICO

Para esta práctica se va a trabajar con un dataset que contiene registros de enfermedades cardiovasculares en distintos pacientes con distintas características como lo son: la edad, el sexo, su IMC (índice de masa corporal), si es diabético, su salud mental, su rendimiento físico, etc. Dicho dataset cuenta con valores de tipo binarios, es decir, el paciente tiene alguna enfermedad cardiovascular “Si/No”.

Lo cual se puede modelar por medio de una regresión logística ya que este tipo de problema se considera “linealmente separable”.

I.DESARROLLO

IMPLEMENTACIÓN EN PYTHON
Para comenzar se importan las librerías correspondientes.

```
import pandas as pd
from sklearn import metrics
import numpy  as np
from sklearn.metrics import *
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
```

Se carga el dataset y se seleccionan las columnas a considerar.

```
dataset = pd.read_csv("dataset.csv")
X=dataset.iloc[:,[1,5]].values
y=dataset.iloc[:,0].values
```

Se codifican las variables categóricas.

```
from sklearn.preprocessing import LabelEncoder
codificadorDatos=LabelEncoder()
y=codificadorDatos.fit_transform(y)
```

Se divide el conjunto de datos en pruebas y entrenamiento.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

Se crea una instancia de LogisticRegression para generar el modelo.

```
clf = LogisticRegression()
```

Se ajusta el modelo al conjunto de entrenamiento.

```
clf = clf.fit(X_train, y_train)
```

Se realizan las predicciones.

```
y_pred = clf.predict(X_test)
```

Se genera una instancia de confusión_matrix para crear la matriz de confusión.

```
cm = confusion_matrix(y_test, y_pred)
print('Matriz de Confusión:')
print(cm)
```

Se genera una tabla con las métricas de la matriz de confusión.

```
print(classification_report(y_test, y_pred))
```

Se calcula la precisión del modelo.

```
accuracy = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:')
print(accuracy)
```

Resultados.

```
Matriz de Confusión:
[[135   0]
 [ 25   0]]
```

	precision	recall	f1-score	support
0	0.84	1.00	0.92	135
1	0.00	0.00	0.00	25
accuracy			0.84	160
macro avg	0.42	0.50	0.46	160
weighted avg	0.71	0.84	0.77	160

```
Exactitud del modelo:
0.84375
```

Como se pudo observar el modelo logro predecir 135 valores de forma acertada y tuvo 25 falsos negativos, junto con una precisión del 84.37 % lo cual supone un modelo “Bueno”.

IMPLEMENTACIÓN EN R

Se cargan los paquetes correspondientes.

```
library(caTools)
library(ISLR)
library(MASS)
library(dplyr)
set.seed(123)
```

Se carga el dataset de enfermedades cardiovascular.

```
dataset = read.csv("dataset.csv")
```

Se codifican las variables categóricas.
Para este caso se van a trabajar con las 15 variables independientes, por lo que se deben codificar las variables categóricas correspondientes.

```
dataset$HeartDisease=factor(dataset$HeartDisease,
                      levels=c("Yes","No"),
                      labels = c(1,0)) #Dummy
dataset$PhysicalActivity=factor(dataset$PhysicalActivity,
                           levels=c("Yes","No"),
                           labels = c(1,0)) #Dummy
dataset$Smoking=factor(dataset$Smoking,
                               levels=c("Yes","No"),
                               labels = c(1,0)) #Dummy
dataset$AlcoholDrinking=factor(dataset$AlcoholDrinking,
                       levels=c("Yes","No"),
                       labels = c(1,0)) #Dummy
dataset$Stroke=factor(dataset$Stroke,
                      levels=c("Yes","No"),
```

```
                        labels = c(1,0)) #Dummy
dataset$DiffWalking=factor(dataset$DiffWalking,
                        levels=c("Yes","No"),
                        labels = c(1,0)) #Dummy
dataset$Sex=factor(dataset$Sex,
                           levels=c("Male","Female"),
                           labels = c(1,0)) #Dummy
dataset$Race=factor(dataset$Race,
                           levels=c("White","Black","Asian","American
Indian/Alaskan Native","Hispanic"),
                           labels = c(0.2,0.4,0.6,0.8,1.0)) #Dummy
dataset$Diabetic=factor(dataset$Diabetic,
                           levels=c("Yes","No","No, borderline diabetes"),
                           labels = c(1,0,0.5)) #Dummy
dataset$GenHealth=factor(dataset$GenHealth,
                           levels=c("Excellent","Very
good","Good","Fair","Poor"),
                           labels = c(1,0.8,0.6,0.4,0.2)) #Dummy
dataset$AgeCategory=factor(dataset$AgeCategory,
                         levels=c("18-24","25-29","30-34","35-39","40-
44","45-49","50-54","55-59","60-64","65-69","70-74","75-79","80 or
older"),
labels=c(0.076,0.158,0.230,0.307,0.384,0.461,0.538,0.615,0.692,0.769,0.846
,0.923,1)) #Dummy
dataset$Asthma=factor(dataset$Asthma,
                         levels=c("Yes","No"),
                         labels = c(1,0)) #Dummy
dataset$KidneyDisease=factor(dataset$KidneyDisease,
                       levels=c("Yes","No"),
```

```
                    labels = c(1,0)) #Dummy
dataset$SkinCancer=factor(dataset$SkinCancer,
                          levels=c("Yes","No"),
                          labels = c(1,0)) #Dummy
```

Se divide el conjunto de datos en 20% prueba y 80% entrenamiento.

```
split = sample.split(dataset$HeartDisease, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
testing_set = subset(dataset, split == FALSE)
```

Se escalan las variables correspondientes ya que el modelo requiere trabajar con valores entre 0 y 1.

```
training_set[c(2,6,7)] = scale(training_set[c(2,6,7)])
testing_set[c(2,6,7)] = scale(testing_set[c(2,6,7)])
```

Se genera una instancia de glm (Generalized Linear Models) indicando que se va a trabajar con un tipo “binomial” para el caso de la regresión logística.

```
regression = glm(formula = HeartDisease ~ .,
                 data = training_set,
                 family = "binomial")
```

Se realizan las predicciones por medio del conjunto de prueba.

```
predicted_values = predict(regression, testing_set, type =
"response")
```

Se determinan los valores de las clases (0 y 1) dependiendo si estos son mayores o menores a 0.5

```
predicted_class = ifelse(predicted_values > 0.5, 1, 0)
performance_data = data.frame(observed = testing_set$HeartDisease,
                              predicted = predicted_class)
```

Se calcula la suma de los valores observados (positivos y negativos).
Posteriormente se calcula la suma de los valores precedidos (positivos y negativos).

```
positive <- sum(performance_data$observed==1)
negative <- sum(performance_data$observed==0)
predicted_positive <- sum(performance_data$predicted==1)
```

```
predicted_negative <- sum(performance_data$predicted==0)
```

Se calcula el total de observaciones.

```
total <- nrow(performance_data)
```

Se genera una tabla de los valores observados y predichos para la matriz de confusión.

```
data.frame(positive,negative,predicted_positive,predicted_
negative)
tp<-sum(performance_data$observed==1 & performance_data$predicted==1)
tn<-sum(performance_data$observed==0 & performance_data$predicted==0)
fp<-sum(performance_data$observed==0 & performance_data$predicted==1)
fn<-sum(performance_data$observed==1 & performance_data$predicted==0)
cm=data.frame(tp,tn,fp,fn)
```

Resultados.

tp	tn	fp	fn
22	133	4	1

Matriz de confusión.

Como se puede observar se obtuvieron en total 155 valores clasificados de forma correcta contra 5 de forma incorrecta lo que resulta en un modelo con una precisión de 96.87% lo que resulta en un muy buen modelo.

IV. CONCLUSIONES

El modelo de regresión logística resulta ser un buen predictor cuando se trabajan con datos de tipo binarios, es decir, cuando los valores de la variable explicada adquieren solo 2 valores (Si/No, 0/1). Sin embargo, este tipo de modelos no resulta ser bueno cuando se trabajan con problemas no lineales, por lo que su utilidad esta limitada a problemas de tipo lineal.

VI. REFERENCIAS

Personal Key Indicators of Heart Disease (2022)
Indicadores clave personales de la enfermedad cardíaca | Kaggle

GeeksforGeeks. (2020, 5 junio). *Logistic Regression in R Programming*. https://www.geeksforgeeks.org/logistic-regression-in-r-programming/

Real Python. (2022, 1 septiembre). *Logistic Regression in Python*. https://realpython.com/logistic-regression-python/