Autor:
Iván García Santillán

Algoritmos predictivos de clasificación: Regresión logística

Introducción

Es un método de aprendizaje automático supervisado utilizado para resolver problemas de clasificación binaria o multinomial (más de dos clases). Aunque su nombre incluye la palabra "regresión", en realidad se trata de un algoritmo de clasificación. Se utiliza comúnmente en una variedad de aplicaciones, como la detección de spam de correo electrónico, diagnósticos médicos, clasificación de documentos --basado en su contenido tales como correo electrónico (spam, no spam), artículo (tecnología, educación, deportes, política, religión), comentario en redes sociales (positivo, negativo o neutral), revisión de productos (satisfecho, insatisfecho), etc.--, entre otros. Por ejemplo, en medicina, es fundamental para predecir la probabilidad de ocurrencia de ciertas enfermedades, como la diabetes o el cáncer, basándose en factores de riesgo. En el sector financiero, se aplica en la evaluación del riesgo crediticio, determinando la probabilidad de que un cliente incumpla con sus pagos. En marketing, se utiliza para predecir la probabilidad de que un cliente compre un producto o responda a una campaña específica. Además, la regresión logística es esencial en el campo de la investigación social y política para analizar la influencia de diversas variables sobre resultados binarios, como la decisión de votar por un candidato. Su capacidad para proporcionar probabilidades y su facilidad de interpretación la hacen una herramienta valiosa para la toma de decisiones basada en datos. Aunque también se debe considerar algunas desventajas como la sensibilidad a datos atípicos (outliers), afectando significativamente la estimación de los coeficientes y distorsionar el modelo. Además, el problema de la multicolinealidad donde hay alta correlación entre las variables independientes (predictoras), así como el problema con datos desbalanceados donde puede desarrollar un sesgo hacia la clase más frecuente.

A continuación, se detallan el funcionamiento del algoritmo y un ejemplo en Python, Scikit-Learn, Keras y TensorFlow para la predicción del cáncer de mama utilizando un data set público.

Contenido

El nombre de Regresión Logística proviene de la función **logística** (también conocida como función **sigmoide**) que utiliza para modelar la relación entre las variables de entrada (características) y la probabilidad de que una instancia pertenezca a una clase específica, como se indica en la Figura 16.

Figura 16. Función logística (sigmoide)

$$f(x) = \frac{1}{1 + e^{-x}}$$

La función logística tiene la siguiente forma:

$$P(Y = 1) = \frac{1}{1+e^{-(a_0+a_1X_1+a_2X_2+\ldots+a_nX_n)}}$$

Donde:

- *P(Y=1)* es la probabilidad de pertenecer a la clase 1.
- *e* es la base del logaritmo natural.

- $a_1, a_2, \ldots, a_n$ son coeficientes que se aprenden (ajustan) durante el entrenamiento, a_0 es el Intercepto o el término constante que también se ajusta junto con los coeficientes.
- $X_1, X_2, \ldots, X_n$ son las variables de entrada.

Durante la fase de entrenamiento, el algoritmo de regresión logística ajusta los coeficientes e intercepto $a_0, a_1, a_2, \ldots, a_n$ de manera que el modelo se ajuste mejor a los datos de entrenamiento.

Una vez que el modelo está entrenado, se puede utilizar para hacer predicciones sobre nuevos datos, es decir, el modelo calcula la probabilidad de que esa instancia pertenezca a una de las clases. Si la probabilidad calculada es mayor que un umbral (generalmente 0.5), se clasifica como la clase positiva (1); de lo contrario, se clasifica como la clase negativa (0) en un problema de clasificación binaria.

A continuación, se presenta un ejemplo del algoritmo de regresión logística en Python, Keras y TensorFlow para la predicción del cáncer de mama (benigno=0, maligno=1) utilizando el conjunto de datos Breast Cancer Wisconsin (Diagnostic) disponible en la librería scikit-learn.

```
# Logistic regression for breast cancer
# importa librerías necesarias
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression

# Carga el conjunto de datos Breast Cancer
dataset = load_breast_cancer()
X = dataset.data # 569x30
y = dataset.target # 569x1

# Divide el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normaliza los datos para que las variables tengan una escala similar
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Crea y entrena el modelo de regresión logistica
model = LogisticRegression(multi_class='auto', solver='lbfgs', max_iter=100)
model.fit(X_train, y_train)

# Imprimir los coeficientes y el intercepto del modelo entrenado
print("Coeficientes del modelo:")
print(model.coef_)
print("Intercepto del modelo:")
print(model.intercept_)

# Realiza predicciones usando el conjunto de prueba
y_pred = model.predict(X_test)

# Convierte las probabilidades en etiquetas binarias (0 o 1)
y_pred = (y_pred > 0.5)

# Muestra el informe de evaluación del modelo entrenado
print(classification_report(y_test, y_pred))

# Matriz de confusión:
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)
print("confusion matrix: \n", cm)
# gráfica de la CM
plt.figure(figsize = (8,4))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Prediction', fontsize = 12)
plt.ylabel('Real', fontsize = 12)
plt.show()

# Exactitud:
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("accuracy: ", acc)

# Sensibilidad:
```

```
from sklearn.metrics import recall_score
recall = recall_score(y_test, y_pred)
print("recall: ", recall)

# Precisión:
from sklearn.metrics import precision_score
precision = precision_score(y_test, y_pred)
print("precision: ", precision)

# Especificidad
# 'specificity' is just a special case of 'recall'.
# specificity is the recall of the negative class
specificity = recall_score(y_test, y_pred, pos_label=0)
print("specificity: ", specificity)

# Puntuación F1:
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("f1 score: ", f1)

# Área bajo la curva:
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("auc: ", auc)

# Curva ROC
from sklearn.metrics import roc_curve
plt.figure()
lw = 2
plt.plot(roc_curve(y_test, y_pred)[0], roc_curve(y_test, y_pred)[1],
color='darkorange',lw=lw, label='ROC curve (area = %0.2f)' %auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# R Score (R^2 coefficient of determination)
from sklearn.metrics import r2_score
```

```
R = r2_score(y_test, y_pred)
print("R2: ", R)
```

Los resultados del modelo de regresión logística se presentan a continuación:

accuracy: 0.97

recall: 0.98

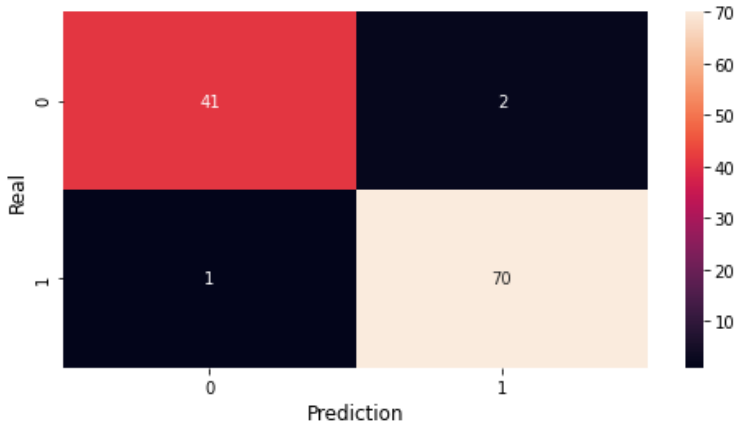
precision: 0.97

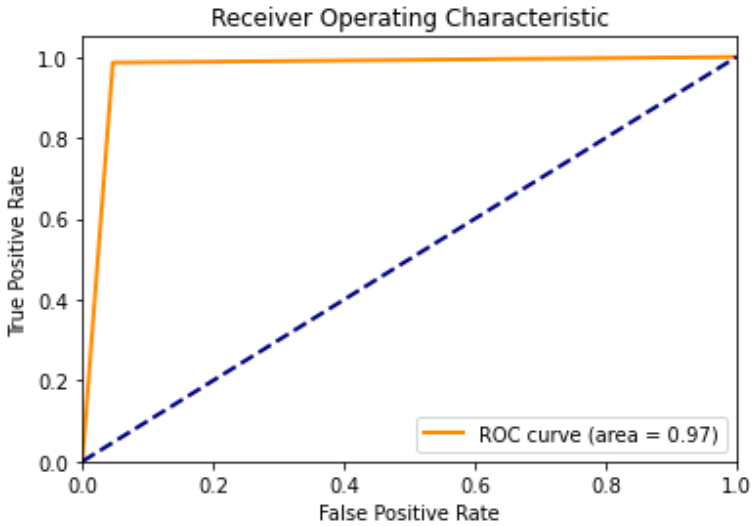
specificity: 0.95

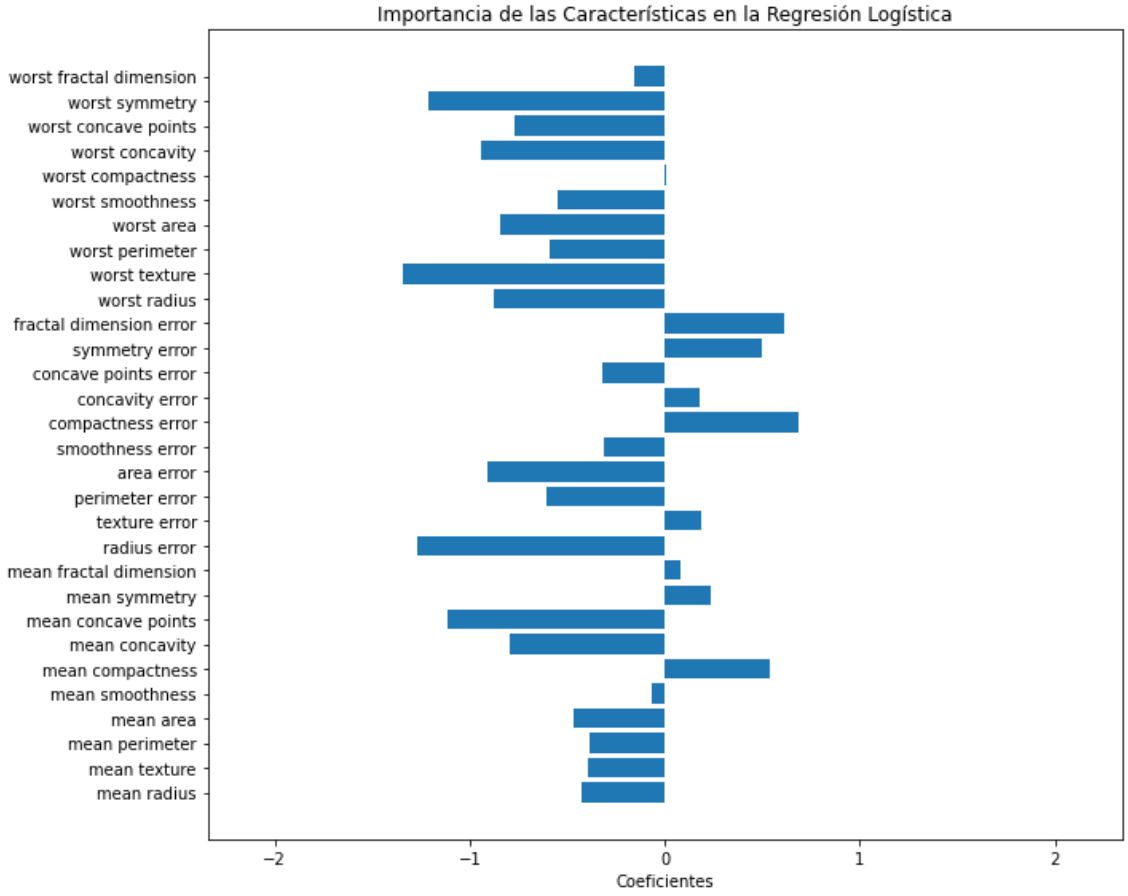
f1 score: 0.97

auc: 0.97

R^2: 0.88







Generalmente, un clasificador adecuado es aquel que alcanza una exactitud (accuracy) superior al 85%. Por lo que, este modelo de regresión logística entrenado cumple con este criterio de calidad.

Los **parámetros** de un modelo de regresión logística son los coeficientes y el intercepto que el modelo aprende durante el proceso de entrenamiento. Los **hiperparámetros** son valores (perillas) que se configuran antes del entrenamiento del modelo y no se ajustan durante el entrenamiento. Algunos de los principales hiperparámetros en la regresión logística son multi_class, solver, y max_iter los cuales son esenciales para configurar el comportamiento del modelo de regresión logística en scikit-learn:

multi_class='auto': Permite que el algoritmo elija la estrategia adecuada para el problema multiclase basado en el solver.

solver='lbfgs': Utiliza el solver L-BFGS, que es un método de optimización de segundo orden eficiente para problemas grandes.
max_iter=100: Limita el número máximo de iteraciones del algoritmo a 100, asegurando que el entrenamiento no continúe indefinidamente.

Adicionalmente, tenga en cuenta que se pueden ajustar varios hiperparámetros del algoritmo (como *multi_class, solver, max_iter, etc.*) para obtener un modelo óptimo. Revise la documentación (ayuda en línea) del algoritmo para más detalles y experimente con ellos. Así mismo, se podría experimentar descartando algunas variables predictoras que no aportan mucho al entrenamiento del modelo basado en un análisis exploratorio de datos como la correlación (entre variables predictoras y con la variable objetivo) o considerando los coeficientes del modelo entrenado de regresión logística (a_0, a_1, a_2, …, a_n).
En una regresión logística, los coeficientes significativos positivos y negativos proporcionan información sobre la relación entre las variables independientes y la probabilidad de que ocurra el evento de interés (la variable dependiente binaria). Los coeficientes del modelo permite interpretar la influencia de cada característica en las predicciones del modelo.
Un coeficiente positivo indica que a medida que aumenta el valor de la variable independiente, la probabilidad de que ocurra el evento de interés también aumenta. Ejemplo: Si el coeficiente de la variable "horas de estudio" es positivo, significa que, a medida que una persona estudia más horas, aumenta la probabilidad de aprobar un examen. La magnitud del coeficiente indica la fuerza de la asociación.
Un coeficiente negativo indica que a medida que aumenta el valor de la variable independiente, la probabilidad de que ocurra el evento de interés disminuye. Ejemplo: Si el coeficiente de la variable "edad" es negativo en un modelo que predice la probabilidad de aprobar un examen, significa que, a mayor edad, menor es la probabilidad de aprobar el examen.
La significancia de los coeficientes se evalúa a través de pruebas estadísticas (como el valor p de la *prueba de Wald*), y solo los coeficientes con valores p menores que un cierto umbral (generalmente 0.05) se consideran significativos (es decir, significativamente diferente de cero).

El gráfico de barras horizontales facilita la comparación visual y ayuda a identificar rápidamente cuáles características tienen un mayor impacto en las predicciones del modelo de regresión logística.

Finalmente, aunque la regresión logística es un modelo fundamental en la caja de herramientas de análisis predictivo, como cualquier otra técnica, presenta tanto ventajas como desventajas que se resumen a continuación:

Ventajas de la Regresión Logística:

Interpretabilidad: Una de las principales fortalezas de la regresión logística es su interpretabilidad fácil y directa. Los coeficientes del modelo pueden ser interpretados para cada uno de los predictores independientes, proporcionando una clara comprensión del impacto de cada variable en la probabilidad del resultado.
Eficiencia computacional: La regresión logística es menos intensiva en términos computacionales en comparación con modelos más complejos como las redes neuronales, lo que la hace relativamente rápida de entrenar.
Buen rendimiento con variables categóricas y numéricas: El algoritmo funciona bien con una combinación de características numéricas y categóricas.
Probabilidades de salida: A diferencia de muchos otros clasificadores, la regresión logística proporciona una probabilidad cuantitativa para las predicciones, lo que puede ser crucial para ciertas decisiones donde el umbral de decisión necesita ser ajustado.
Bueno para modelos lineales: Funciona muy bien si la relación entre la variable dependiente y las independientes es aproximadamente lineal, común en problemas de complejidad baja y media.

Desventajas de la Regresión Logística:

No es adecuada para relaciones no lineales: El algoritmo no puede manejar relaciones no lineales sin transformación de las variables independientes, común en problemas complejos. El término modelar "relaciones no lineales" se refiere a la capacidad de un modelo estadístico o de aprendizaje automático para capturar y representar relaciones entre variables que no son simplemente proporcionales

o directas. En una **relación no lineal**, los cambios en una variable no producen cambios proporcionales y predecibles en otra variable, lo que significa que la relación entre variables no puede ser descrita adecuadamente por una línea recta, sino por otro tipo de relación como cuadrática, exponencial, logarítmica, sinusoidal, etc.

Sensibilidad a datos no balanceados: Si las clases no están aproximadamente igualmente representadas, la regresión logística puede desarrollar un sesgo hacia la clase dominante. Esto puede requerir técnicas de muestreo o ponderación para corregir.

Influencia de valores atípicos: Como en muchos modelos lineales, los valores atípicos pueden tener un efecto desproporcionadamente alto en la estimación del modelo, lo que puede llevar a estimaciones y predicciones pobres.

Multicolinealidad: Problemas de multicolinealidad pueden surgir cuando dos o más variables predictoras son altamente correlacionadas. Esto no solo influye en la precisión de la estimación de los coeficientes, sino también en la interpretación de estos.

Conclusiones

En esta lectura se ha revisado los fundamentos del algoritmo de Regresión Logística para abordar problemas de clasificación binaria utilizando Python y TensorFlow-Keras. La regresión logística es una herramienta poderosa y flexible para problemas de clasificación, especialmente cuando la claridad de la interpretación y la velocidad de cálculo son preocupaciones primordiales. Sin embargo, su aplicación debe ser considerada viendo que las limitaciones del modelo no afecten negativamente los resultados del análisis. Es importante experimentar y ajustar varios hiperparámetros del algoritmo para obtener un modelo óptimo.

Por lo que, se motiva a los estudiantes a revisar material y ejemplos adicionales disponibles libremente en Internet para profundizar en la comprensión del algoritmo y su aplicación eficaz en diferentes problemas de variada complejidad.

Bibliografía:

Berzal, Fernando (2019). Redes neuronales & Deep Learning. USA: Independently published.

Gerón, A. (2023). Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow: Conceptos, herramientas y técnicas para conseguir sistemas inteligentes. Tercera Edición. Anaya Multimedia.

Torres, J. (2020). Python Deep Learning: Introducción práctica con Keras y TensorFlow 2. Marcombo.