Fecha 9 Septiembre 2022

Tecnológico de Monterrey Campus Querétaro

Desempeño de un modelo de regresión logística. (Septiembre, 2022)

Castro Payns, Mariana. A01706038

ABSTRACT Dentro del presente documento se realizará el análisis del desempeño de un modelo de regresión logística implementado con ayuda de frameworks encontrados en la librería "Scikit-Learn" haciendo uso del dataset de Iris. Dicho modelo realiza la predicción de la clase de una planta Iris.

PALABRAS CLAVE Regresión logística, Scikit-Learn, Iris, Clase.

I. INTRODUCCIÓN

El machine learning es una disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, da la capacidad de identificar patrones en datos masivos para con ellos elaborar predicciones, uno de los métodos más comunes para esto son regresiones gracias a que es fácil de implementar y entender y proporciona buenos resultados para una gran variedad de problemas.

El método utilizado para la predicción buscada fue un modelo de regresión logística. Una regresión logística es un tipo de análisis de regresión usado en diferentes áreas para predecir el resultado de una variable de tipo categórica.

Dicho esto, podemos decir que la selección de este tipo de regresión fue debido a que el propósito del modelo es predecir una clase de la manera más precisa posible.

Para este caso la predicción dirá la clase de un tipo de planta iris tomando como referencia o bien, como datos de entrada variables de las mismas plantas, como lo son el alto y ancho de pétalos y sépalos.

II. DATASET

El dataset utilizado se encuentra en el repositorio de Machine Learning de la UCI, es llamado Iris fue creado por R.A Fisher en el año de 1936 y tiene características multivariables y sus atributos son reales, por otro lado tiene 150 instancias y 4 atributos los cuales son alto del sépalo, ancho del sépalo, alto de pétalo y ancho del pétalo.

Las clases del dataset son 'Iris Setosa", "Iris Versicolour" y "Iris Viginica".

Es importante mencionar que el dataset no tiene datos faltantes.

Al ver el dataset (Figura 1.), es posible ver que se tiene una columna doble de ID, esto es porque el dataset tenía incluída una columna de ID, pero la librería Pandas, que es usada para cargar los datos crea dicha columna por default, es por esto que la columna no resulta relevante y se cortó del dataset para evitar que causara problemas al momento del entrenamiento del modelo.

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

FIGURE 1. **Iris Dataset**

III. MODELO

Como se ha mencionado anteriormente se implementó un modelo de regresión logística.

Para la realización de este modelo primero fue necesario importar las librerías necesarias que en este caso fueron "Numpy", "Pandas", "Scikit-Learn" y "matplotlib".

Seguido de esto se dividió en dataset para obtener train, test y validación, después especificó el tipo de modelo y se realizó el entrenamiento con los datos previamente divididos.

Hecho el modelo se determinaron factores como los parámetros del modelo, el bias, una matriz de confusión, su porcentaje de precisión así como el método de cross validation, esto con el fin de comprender el funcionamiento del modelo en cuanto a su sesgo y varianza y ajuste.

De igual manera se realizaron múltiples predicciones para comprobar el funcionamiento del modelo bajo diferentes condiciones.

IV. SEPARACIÓN DE LOS DATOS

Los datos fueron separados para el mejor aprovechamiento de estos, se separan en tres grupos test, train y validation. Esto con el fin de minimizar posibles sesgos del modelo, ya que no tiene acceso a los datos de prueba debido a que estos se utilizan para medir el desempeño del modelo, mientras que los datos de entrenamiento y validación se utilizan para el aprendizaje del modelo.

Así mismo el proceso se realiza con el propósito de evitar que el modelo "sobre aprenda" o bien tenga problemas de overfitting y al mismo tiempo para evaluar el modelo de forma más efectiva.

El set de train es usado para entrenar al modelo, y que este aprenda; el set de validación es usado para proporcionar una evaluación imparcial de un modelo ajustado en el conjunto de datos de entrenamiento mientras se ajustan los hiper parámetros del modelo; el set de test es empleado para evaluar al modelo final ajustado en el set de datos de entrenamiento.

Este proceso se realizó en el código mostrado a continuación con ayuda de la librería (Figura 2):

```
# Split data into train and test
Xtrain, Xtest, ytrain, ytest = train_test_split(X, exp, test_size=0.2)

# Split data to get a validation set
Xtrain, Xval, ytrain, yval = train_test_split(Xtrain, ytrain, test_size=0.25)
```

FIGURE 2. **Código separación de datos.**

Dentro del código podemos observar que primero se divide el dataset original con los valores en y o bien los esperados, en train y test con un 20%; una vez que se hizo dicha separación se toma el set ya separado de train y se divide en un 25% para obtener el set de validación.

El tamaño de los sets una vez hecha la separación se puede ver en la figura 3, el data frame original y el expected o Y tienen un tamaño de 150 instancias, una vez que se separó, train quedó con 90 instancias, y test y validación con 30.

```
Size of DF -  (150, 4)
Size of expected values -  (150,)
X train shape:  (90, 4)
y train shape:  (90,)
X test shape:  (30, 4)
y test shape:  (30,)
X val shape:  (30, 4)
y val shape:  (30, 4)
```

FIGURE 3. **Impresión de tamaños de los sets de datos.**

V. RESULTADOS

Después de que se entrenó el modelo, lo siguiente sería evaluar su desempeño por medio de las predicciones que arrojó el modelo.

La librería "Scikit-Learn" da un accuracy por default, para este modelo dicho accuracy fue de 0.96667, lo cual indica una precisión de 96% en las predicciones.

Sin embargo, para poder determinar el desempeño de un modelo de una manera más completa no es suficiente basarse únicamente en el accuracy, es por ello que se obtuvo una matriz de confusión, esta es una forma tabular de visualizar el rendimiento de las predicciones, en donde cada entrada de la matriz denota el número de predicciones donde se clasificó como verdadero-positivo, falso-positivo, falso-negativo y verdadero-negativo.

Para este caso la matriz es de multiclase (Figura 4), dicha matriz tiene un total de 29 positivos y 1 negativo.

```
Confusion Matrix:
[[ 9  0  0]
 [ 0 12  0]
 [ 0  1  8]]
```

FIGURE 4. **Matriz de confusión.**

Por otro lado, si vemos la gráfica del accuracy del modelo de training y validation (Figura 5), obtenida como curva de aprendizaje, es posible ver que la validación del modelo logró aumentar su aprendizaje de manera significativa.

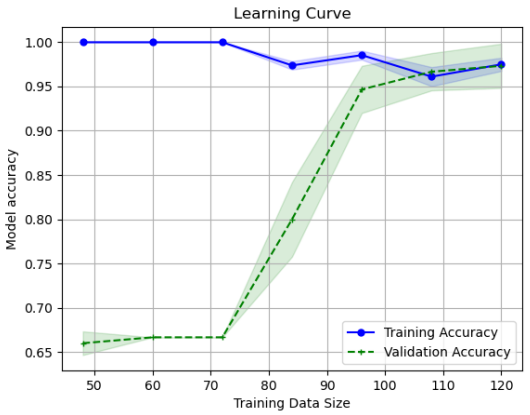

FIGURE 5. **Learning Curve.**

VI. NIVEL DE AJUSTE DEL MODELO

Para poder determinar el nivel de ajuste del modelo, es necesario ver la gráfica de la curva de aprendizaje (Figura 5), al analizar este tipo de gráficas buscamos que ambos sets se parezcan lo más posible.

Cuando el training accuracy es bajo al igual que el validation accuracy se tiene un problema de underfitting, por otro lado si se tiene un alto nivel de training accuracy pero un nivel de validation accuracy bajo, indica overfitting.
En este caso al ver la gráfica, podría parecer a primera instancia que el modelo tiene overfitting, ya que inicia con un valor de accuracy de validación muy bajo, sin embargo conforme avanza el dataset, la línea de validation accuracy aumenta considerablemente hasta que alcanza a la línea del training accuracy, logrando que ambas líneas se junten y tengan comportamientos similares. Ahora, la sombra alrededor de las líneas indica el error de cada uno de los sets, en ninguno de los casos es significativo.
Por otro lado, al hacer el una media del cross validation, el cual sirve para determinar situaciones de niveles de ajuste, se obtuvo un valor de 0.973333 para test, el cual indica un 97%.
Mencionado lo anterior es posible decir que el modelo no se encuentra en estado de underfitting ni overfitting.

VII. DETECCIÓN DE SESGO Y VARIANZA

Dado el hecho de que el accuracy fue alto y analizando la gráfica, es posible decir que el modelo tiene sesgo bajo así como una varianza baja, ya que se tiene un nivel de error bajo en las predicciones, dichos errores se pueden observar en la gráfica (Figura 5) en la línea sombreada, como podemos ver los errores no son significativos. Esto también nos ayuda a confirmar el nivel del modelo, siendo que no se encuentra en estado de underfitting ni overfitting.

VIII. REGULARIZACIÓN O AJUSTE

En el modelo realizado se obtuvieron los resultados esperados, lo cual quiere decir que no es necesario realizar ningún tipo de ajuste, sin embargo, es posible mejorar el modelo para que se tenga un mejor desempeño y a su vez una mejor accuracy. Esto se puede hacer con diferentes técnicas como lo son la regularización.
De igual forma cabe mencionar, que sería necesario tomar en cuenta funciones de activación, como lo son la sigmoide.

REFERENCIAS:

Agraval, S (2021). “Split data in three sets”. Recuperado el 9 de septiembre de 2022 del sitio web Towards Data Science: https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c

Mohajón, J (2020). “Confusion matriz for a multi-class machine learning model”. Recuperado el 8 de septiembre de 2022 del sitio web Towards Data Science: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826

UCI (2022). “Iris Data Set”. Recuperado el 9 de septiembre de 2022 del sitio web: https://archive.ics.uci.edu/ml/datasets/Iris