TP2 Reconocimiento de Patrones

Corvalan César, Junqueras Juan, Suárez Juan Carlos

Junio 2021

Introducción

En el presente trabajo práctico se utiliza el modelo de regresión Logistic Regression como método de clasificación del dataset Iris.csv.

1. Marco teórico

La regresión logística es una técnica analítica que nos permite relacionar funcionalmente una variable discreta (dicotómica ó tricotómica, ó más) con un conjunto de variables independientes. La regresión logística puede considerarse una extensión de los modelos de regresión lineal, con la particularidad de que el dominio de salida de la función está acotado al intervalo [0,1] y que el procedimiento de estimación, en lugar de mínimos cuadrados, utiliza el procedimiento de estimación máximo-verosímil.
Regresión Logística lleva el nombre de la función utilizada en el núcleo del método, la función logística, también llamada función Sigmoide. Esta función es una curva en forma de S que puede tomar cualquier número de valor real y asignar a un valor entre 0 y 1.

$$P(z) = \frac{1}{1 + e^{-z}}$$

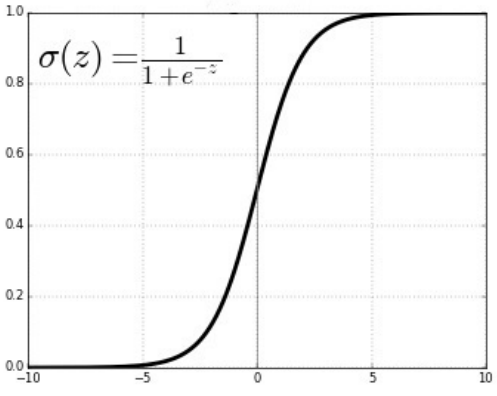


Figura 1: Función Sigmoide

La Regresión Logística es uno de los algoritmos de Machine Learning más utilizados para la clasificación de clases. Describe y estima la relación entre una variable generalmente binaria (en este informe ternaria) dependiente y las variables independientes. Veamos de que manera lo logra.

Definimos la representación del hiperplano como

$$w_0 + \sum_{i=1}^{n} (w_i x_i) = 0$$

Donde

$$w = (w_0, w_1, ..., w_n)^T$$
$$x = (1, x_1, ..., x_n)^T$$

Forma compacta

$$w^T x = 0$$

En la siguiente imagen podemos ver que la recta, representada con color azul, obtenida mediante regresión lineal funciona como un clasificador bidimensional, por un lado aquellos puntos que se encuentran por encima de la recta y los que se encuentran por debajo. Se define también un umbral, de color verde y los diferentes vectores descritos (puntos en el plano). Como se ve, existen cuatro regiones claramente definidas, aquellas que se encuentran dentro del umbral, o fuera de el, tanto por encima como por debajo de la frontera, adicionalmente, puede suceder que el vector descriptor coincida con cualquiera de las rectas trazadas, decidir el comportamiento o la respuesta del clasificador cuando el vector cae en cualquiera de las circunstancias descritas, es meramente una cuestión de diseño, puede tratarse de una respuesta positiva, negativa o indeterminada

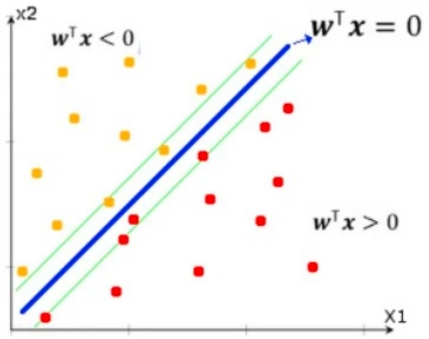


Figura 2: Plano clasificador

El objetivo es calcular la probabilidad de una variable cualitativa en función de una variable cuantitativa, es decir, la probabilidad de que un punto del plano se ubique por arriba o por debajo de la recta frontera.
Si una variable cualitativa con dos niveles se codifica como 1 y 0, matemáticamente es posible ajustar un modelo de regresión lineal por mínimos cuadrados $Y = w^T x$. El problema de esta aproximación es que, al tratarse de una recta ó plano, para valores extremos del predictor, se obtienen valores de Y menores que 0 o mayores que 1, lo que entra en contradicción con el hecho de que las probabilidades siempre están dentro del rango [0,1].
Para evitar estos problemas, la regresión logística transforma el valor devuelto por la regresión lineal empleando una función cuyo resultado está siempre comprendido entre 0 y 1. La función Sigmoide que definimos al principio.

$$P(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

1.1. Descripción del problema

El problema a resolver consiste en utilizar Regresión Logística para clasificar los tipos de flores Iris que figuran en el Iris dataset.

2. Desarrollo

Uno de los primeros pasos a la hora de realizar un proyecto que involucre el análisis de datos es explorar y visualizar los datos. El objetivo principal es obtener información sobre el contenido del dataset.

2.1. Análisis del dataset

Cantidad de filas y columnas

```
print("\n", dataset.shape) # Number of rows and columns
```

```
(150,5)
```

El archivo consta de 150 filas y 5 columnas.

Imprimimos los primeros 15 registros, para visualizar la información del dataset.

```
print("\n", dataset.head(15)) # Select first 15 rows to take a peek at the data
```

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa

Podemos ver que cada registro del dataset posee información sobre el largo y ancho del sépalo, el largo y el ancho del pétalo y la clase de Iris a la cual pertenecen los valores de dicho registro.

Imprimimos datos estadísticos del dataset.

```
print("\n", dataset.describe()) # Summary statistics of data
```

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Analizamos cuales y cuantas clases posee el dataset.

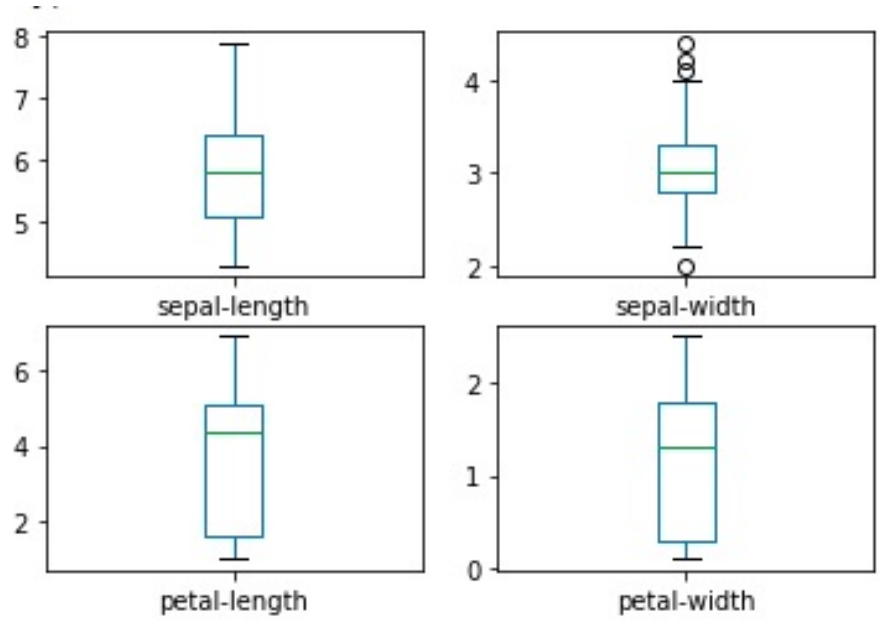
```
print("\n", dataset.groupby('class').size()) # Get rows in each class
```

```
 class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

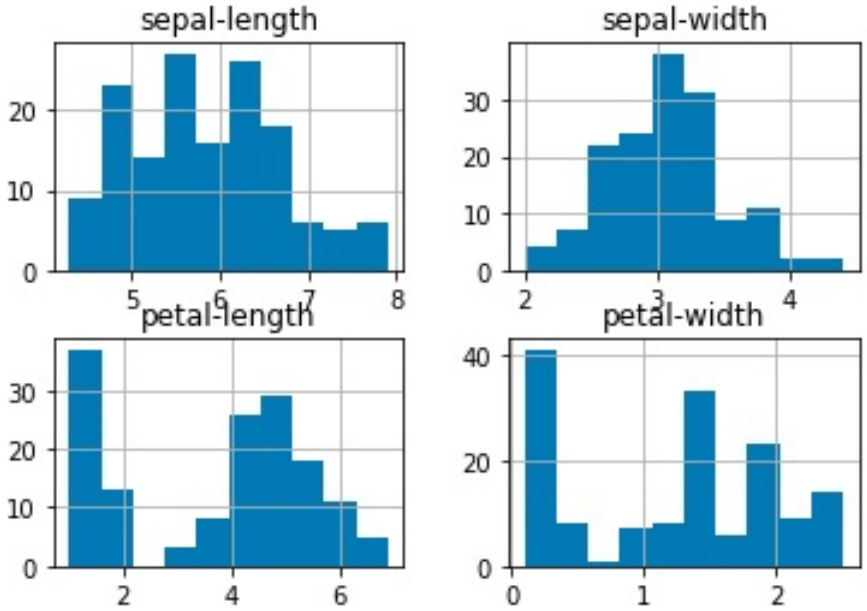
Aqui podemos verificar que los datos corresponden a tres clases de Iris: Iris setosa, Iris versicolor e Iris virginica. El dataset consta de 50 registros con información sobre las dimensiones del sépalo y pétalos para cada una de las clases y están expresadas en formato Int de 64 bits.

Veamos unos gráficos que nos permitirán visualizar la distribución de los datos.

```
# Visualise the data
# box plot
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

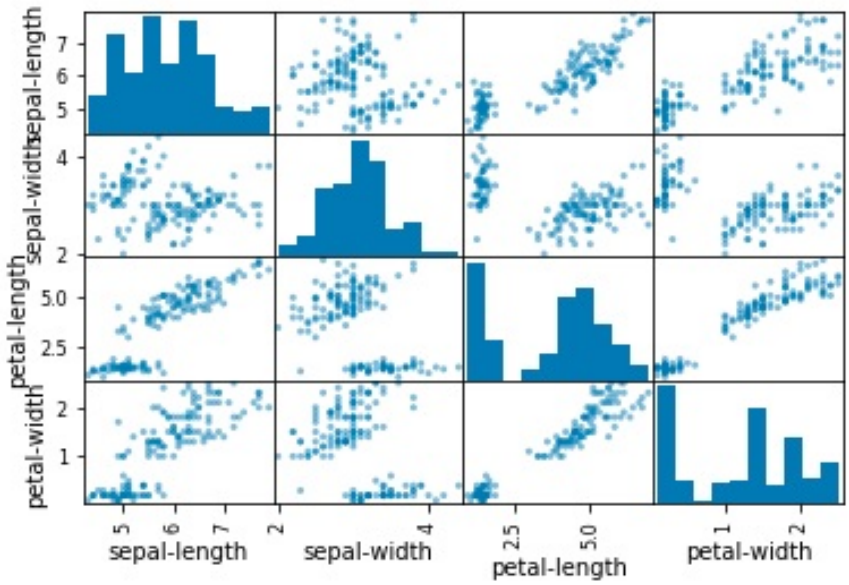



```
# histogram
dataset.hist()
plt.show()
```



Una características que puede verse a simple vista es que los valores de las dimensiones de los sépalos tienen una distribución similar a una normal, ellos "dibujan" una curva similar a una campana de Gauss. En cambio las dimensiones de los pétalos parecen concentrarse en dos grupos visiblemente disjuntos. En principio podemos inferir que las dimensiones de los pétalos pueden ser muy diferentes entre clases distintas de Iris.

```
# scatter plot matrix
scatter_matrix(dataset)
plt.show()
```



Si analizamos la matriz de dispersión no parece haber relación entre las dimensiones de los sépalos ni entre las dimensiones de los sépalos con los pétalos. En cambio, si parece existir una fuerte relación entre las dimensiones de los pétalos, cuanto más largos más anchos.

Una características que puede verse a simple vista es que los valores de las dimensiones de los sépalos tienen una distribución similar a una normal, ellos "dibujan" una curva similar a una campana de Gauss. En cambio, las dimensiones de los pétalos parecen concentrarse en dos grupos visiblemente disjuntos. En principio podemos inferir que las dimensiones de los pétalos pueden ser muy diferentes entre clases distintas de Iris.

Para poder realizar regresión logística, primero vamos a almacenar en un arreglo llamado X los datos correspondientes a las dimensiones de los sépalos y pétalos, y en otro arreglo llamado Y las clases de Iris correspondientes. Notar que separamos un 80 % de los datos para entrenamiento y el 20 % restante para testear.

```
# Evaluate algorithms
# Get test and validation set
array = dataset.values
X = array[:,0:4] # Select Sepal and Petal lengths and widths
Y = array[:,4] # Select class
testing_size = 0.20
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size = testing_size, random_state = seed)
```

Para analizar la performances vamos a compararlo con otros métodos de regresión. Realizamos cross validation con cada uno de los métodos e imprimimos la media de precisión (accuracy) y el desvío standar de dicha precisión

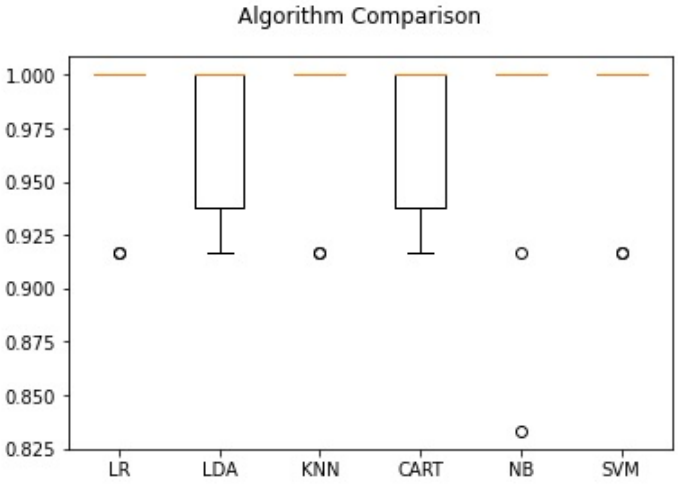
```
# Spot check algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

# Evaluate each model in turn
results = []
names = []
kFoldsplits = 10
print("")

for name, model in models:
    kfold = model_selection.KFold(n_splits = kFoldsplits, random_state = seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    print("{}:     {}:     ({})".format(name, format(cv_results.mean(), '.3f'), format(cv_results.std(), '.3f')))
```

Podemos ver que junto a otros dos métodos la regresión logística es uno de los más precisos.

```
LR:     0.983:     (0.033)
LDA:     0.975:     (0.038)
KNN:     0.983:     (0.033)
CART:     0.975:     (0.038)
NB:     0.975:     (0.053)
SVM:     0.983:     (0.033)
```




```
LR = LogisticRegression()
LR.fit(X_train, Y_train)
predictions = LR.predict(X_test)
acc_score = accuracy_score(Y_test, predictions)

print("\n", "Accuracy of LR model: {}".format(format(acc_score,'.3f')))
print("\n", "Confusion matrix: \n", confusion_matrix(Y_test, predictions))
print("\n", "Classification report: \n", classification_report(Y_test, predictions))
```

Luego de entrenar el modelo, y aplicarlo a los datos de test obtenemos que predice las clases de Iris con un 86,7 % de acierto. Pero en la "confusion matrix" y en el "reporte de clasificación" podemos analizar con qué efectividad predijo cada clase en particular.

- Iris Setosa, acertó en 7 de 7 casos (100 %).
- Iris Versicolor, acertó en 10 de 12 casos (83 %).
- Iris Virginica, acertó en 9 de 11 casos. (82 %).

```
Accuracy of LR model: 0.867

Confusion matrix:
[[ 7  0  0]
 [ 0 10  2]
 [ 0  2  9]]

Classification report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.83      0.83      0.83        12
 Iris-virginica       0.82      0.82      0.82        11

       accuracy                           0.87        30
      macro avg       0.88      0.88      0.88        30
   weighted avg       0.87      0.87      0.87        30
```

3. Conclusiones

Al comparar la regresión logística con otros métodos obtuvimos que LR es, al menos, tan eficiente como los otros para clasificar las clases de Iris. Otra observación importante es lo bien que funciona el método cuando el dataset posee clases muy diferenciadas entre si, o al menos tienen una característica que las diferencia (la dimensión de los pétalos), como es el caso de la clase Iris Setota, las dimensiones de los pétalos de este tipo de Iris son muy diferentes a las de las otras clases y por ende, el método acertó el 100 % de las veces. En cambio, en aquellas clases que pueden llegar a superponerse en algunos puntos (el concepto de umbral que mencionamos en el marco teórico), es decir, casos de sépalos y pétalos con dimensiones similares en ambas clases, como ocurre con la Iris Versicolor y la Iris Virginica, el método obtuvo una eficacia levemente superior al 80 %. Catalogar a un método de clasificación como bueno o malo va a depender del nivel de tolerancia que tengamos para un dataset particular. En algunos ambitos un 80 % de eficacia en la predicción puede ser aceptable y en otros contextos no.