Entrega Módulo 4 BDG:

“Machine Learning”

1. Obtenga (analíticamente) el F1 score en función de TP, TN, FP, FN. **(0.5p)**

Nota: Es posible resolver este ejercicio en papel, adjuntando una captura fotográfica de los cálculos.

2. Un clasificador multiclase ha obtenido la siguiente matriz de confusión en la evaluación post-entrenamiento, cada clase teniendo el mismo soporte: **(3p)**

		Clase real					
		N	L	R	A	P	V
Clase predicha	N	1945	25	8	10	7	5
	L	14	1951	3	21	7	4
	R	17	11	1845	87	19	21
	A	11	27	137	1790	16	19
	P	0	42	4	49	1899	6
	V	3	12	5	5	7	1968

Calcule las siguientes métricas de rendimiento: precisión (***accuracy***), exactitud (***precision***), exhaustividad (***recall***) y **F1 score** (utilizando la formula obtenida en el ejercicio 1) del clasificador entrenado.

Estas métricas se obtendrán a partir de los valores TP, TN, FP, FN (encontradas en la matriz de confusión), tanto a nivel de cada clase (N, L, R, A, P y V), como a nivel global.

Autor: Izan González Panadero

1. Obtenga (analíticamente) el F1 score en función de TP, TN, FP, FN.

$$F = 2 \cdot \frac{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} = \frac{2 \cdot TP}{FN + FP + 2TP}$$

2. Clasificación

Para N
FP = 55 TP = 2945
FN = 45 TN = 9453
A = 0,9913 P = 0,9725 R = 0,9773 F1 = 0,9749

Para L
FP = 49 TP = 2951
FN = 117 TN = 9447
A = 0,9856 P = 0,9755 R = 0,9434 F1 = 0,9591

Para R
FP = 156 TP = 1841
FN = 157 TN = 9553
A = 0,9733 P = 0,9725 R = 0,9215 F1 = 0,9220

Para A
FP = 210 TP = 1790
FN = 172 TN = 9608
A = 0,9675 P = 0,895 R = 0,9123 F1 = 0,9035

Para P
FP = 101 TP = 1899
FN = 66 TN = 9499
A = 0,9864 P = 0,9495 R = 0,9718 F1 = 0,9605

Para V
FP = 32 TP = 1968
FN = 55 TN = 9430
A = 0,9924 P = 0,984 R = 0,9728 F1 = 0,9783

Para Gbbal
FP = 258 TP = 11398
FN = 344 TN = 11398
A = 0,9498 P = 0,9778 R = 0,9707 F1 = 0,9742

Autor: Izan González Panadero

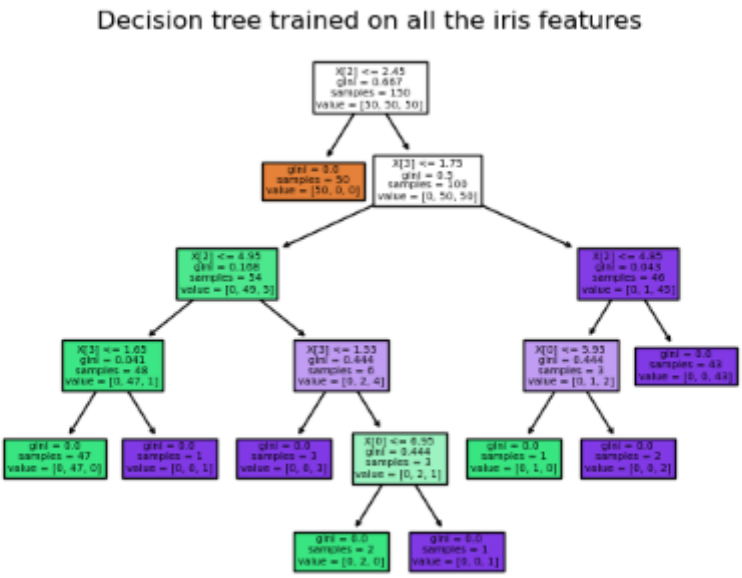
4. **Describa** el algoritmo que ha obtenido las mejores métricas de rendimiento en el ejercicio 3 (2-4 páginas, incluyendo bibliografía). Esta descripción incluirá:
    a. Desafíos solucionados por el algoritmo. Uso y aplicaciones.
    b. Una explicación simplificada de su funcionamiento desde el punto de vista matemático.
    c. Una revisión de su implementación en `scikit-learn` (revisión de la API, exploración de los otros posibles parámetros).

Al observar que los datos debían tener algún tipo de tratamiento previo y generar en ocasiones métricas de rendimiento perfectas según el randon state que seleccionemos no se puede decir con exactitud cual es el mejor algoritmo de clasificación para estos datos, por lo que paso a describir el algoritmo de árboles de decisión que es el que ha dado los resultados mas parecidos entre si en las distintas clasificaciones realizadas con el dataset.

Decision Tree o Árboles de Decisión

Es un algoritmo de clasificación, el cual lleva crea un modelo que prediga el valor de una variable destino mediante una aproximación constante por pares.

Uno de los beneficios de esta forma de clasificación es que es fácil de entender, además se puede visualizar y resultar más intuitiva.



Suele ser usada en datos que no se tengan preparados previamente, a excepción de comprobar que no falte ningún valor.

En contraposición una desventaja de este método es que nos puede generar un sobre ajuste en los datos más fácilmente que otras formas de clasificación, solucionándose mediante la

elección de un número mínimo y uno máximo dentro de los datos e iteraciones que lleva a cabo este método.

Los arboles de decisión suelen clasificar llevando a cabo el algoritmo de Hunt, el cual se basa en la división en subconjuntos que buscan una separación óptima, se dividen los datos en subconjuntos más pequeños en función de una variable y se repite el proceso.

Para decidir que variable usar se consideran el **Error de Clasificación, el índice Gini (rpart) o la Entropía (C50).**

Se define el **índice de Gini** como:

$$GINI(t) = 1 - \sum_{i=1}^{n} (P_i)^2$$

Donde Pi es la probabilidad de que un ejemplo sea de la clase i.

Se define la **entropía** como:

$$H = -\sum_{i=1}^{n} P_i * \log_2 P_i$$

Donde Pi es la probabilidad de que un ejemplo sea de la clase i.

Se define el **RSS** como:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Donde yi es el valor real de la variable a predecir y ^yi es el valor predicho.

Ejemplo de implementación

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

- https://en.wikipedia.org/wiki/Decision_tree_learning
- https://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20(DTs)%20are%20a,as%20a%20piecewise%20constant%20approximation.
- https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/

Autor: Izan González Panadero