



**UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES**

Centro de Ciencias Básicas

Ciencias de la Computación

Aprendizaje Inteligente

Trabajo 5.

Ingeniería en Computación Inteligente semestre 6°A

Profesor: Francisco Javier Luna Rosas

Alumnos:

José Luis Cardona Rivera.

José Horacio Cervantes Palacios.

Dataset:

El dataset consta de imágenes de los dígitos 0 y 1, que han sido descompuestas por píxeles, obteniendo así el data set. Todos estos píxeles son las características de los dígitos, estas características serán nuestras variables predictoras. Nuestras variables a predecir es la clasificación que se le dará a cada conjunto de píxeles, para decirnos si es un número 0 o un 1 .

1	pixel197	pixel198	pixel199	pixel200	pixel201	pixel202	pixel203	pixel204	pixel205	pixel206	pixel207	pixel208	pixel209	pixel210	pixel211	pixel212	pixel213
2	0	0	0	0	0	0	0	0	213	253	253	253	253	253	253	253	253
3	0	0	0	0	0	0	0	0	107	128	168	250	250	250	252	250	250
4	0	0	0	0	0	0	0	0	0	0	0	26	181	233	102	40	40
5	0	0	0	0	0	0	0	0	0	0	0	40	0	2	12	45	45
6	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	20	20
7	0	0	0	0	0	73	224	254	254	156	156	89	59	14	0	0	0
8	0	0	0	0	0	0	0	0	0	0	5	117	251	251	243	212	212
9	0	0	0	0	0	0	0	0	0	32	193	146	40	0	0	89	89
10	0	0	0	0	0	0	0	0	0	96	189	251	251	253	251	251	251
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	155	253	253	253	253	253	253	253
13	0	0	0	0	0	0	0	0	15	21	57	252	252	253	189	210	210
14	0	0	0	0	0	0	157	253	194	9	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	191	255	255	255	255	255	255	255	255
16	0	0	0	0	0	0	0	128	255	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	127	252	252	253	252	252	252	252	253	252	252
18	0	0	0	14	228	253	200	86	8	0	0	0	60	253	129	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	78	253	0	0	0
20	0	0	0	0	0	0	0	0	0	114	253	251	253	251	253	251	251
21	0	0	0	0	0	0	0	0	132	252	233	70	151	50	51	232	232

Algoritmos usados funcionamientos:

1: Red Neuronal (MLPClassifier) Perceptron multicapa clasificador:

MLPClassifier significa clasificador perceptrón multicapa que en el propio nombre se conecta a una red neuronal. A diferencia de otros algoritmos de clasificación MLPClassifier se basa en una red neuronal subyacente para realizar la tarea de clasificación.

Un perceptrón multicapa (MLP) es una red neuronal profunda y artificial. Se compone de más de un perceptrón. Se componen de una capa de entrada para recibir la señal, una capa de salida que toma una decisión o predicción sobre la entrada, y entre esas dos, un número arbitrario de capas ocultas que son el verdadero motor computacional del MLP. Los MLPs con una capa oculta son capaces de aproximar cualquier función continua.

Si un perceptrón multicapa tiene una función de activación lineal en todas las neuronas, es decir, una función lineal que asigna las entradas ponderadas a la

salida de cada neurona, entonces el álgebra lineal muestra que cualquier número de capas se puede reducir a un modelo de entrada-salida de dos capas. En mlps algunas neuronas utilizan una función de activación no lineal que fue desarrollada para modelar la frecuencia de los potenciales de acción, o disparo, de las neuronas biológicas.

Las dos funciones de activación históricamente comunes son ambas sigmoideas, y son descritas por:

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}.$$

El aprendizaje se produce en el perceptrón cambiando los pesos de conexión después de procesar cada dato, en función de la cantidad de error en la salida en comparación con el resultado esperado. Este es un ejemplo de aprendizaje supervisado, y se lleva a cabo a través de la retropropagación, una generalización del algoritmo de mínimos cuadrados medios en el perceptrón lineal.

Podemos representar el grado de error en un nodo de salida j en el n -ésimo punto de datos (ejemplo de entrenamiento) por $e_j(n) = d_j(n) - y_j(n)$, donde d es el valor de destino y y es el valor producido por el perceptrón.

2: Árboles de decisión:

Los árboles de decisión (DTs) son un método de aprendizaje supervisado no paramétrico utilizado para la clasificación y regresión. El objetivo es crear un modelo que prediga el valor de una variable de destino mediante el aprendizaje de reglas de decisión simples inferidas de las características de datos. Un árbol se puede ver como una aproximación constante por partes.

Dados los vectores de entrenamiento x_i , $i = 1, \dots, l$ y un vector de etiqueta y , un árbol de decisión divide recursivamente el espacio de entidades de modo que las muestras con las mismas etiquetas o valores de destino similares se agrupan.

Deje que los datos en el nodo se representa con ejemplos. Para cada división candidata que consiste en una característica y un umbral, particione los datos en y subconjuntos

$$Q_m, N_m, \theta = (j, t_m) \quad Q_m^{left}(\theta), Q_m^{right}(\theta)$$

$$Q_m^{left}(\theta) = \{(x, y) | x_j \leq t_m\}$$

$$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$$

La calidad de una división candidata de nodo se calcula utilizando una función de impureza o función de pérdida, cuya elección depende de la tarea que se está resolviendo (clasificación o regresión) $H()$

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} H(Q_m^{right}(\theta))$$

Seleccione los parámetros que minimizan la impureza

$$\theta^* = \operatorname{argmin}_{\theta} G(Q_m, \theta)$$

Recursar para subconjuntos y hasta que se alcance la profundidad máxima permitida, o

$$Q_m^{left}(\theta^*) \cup Q_m^{right}(\theta^*) N_m < \min_{samples}$$

$$N_m = 1$$

3. KNN (K Neares Neighbor) K vecinos más cercanos

K-Nearest-Neighbor es un algoritmo basado en instancia de tipo supervisado de Machine Learning. Puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Sirve esencialmente para clasificar valores buscando los puntos de datos “más similares” (por cercanía) aprendidos en la etapa de entrenamiento y haciendo conjeturas de nuevos puntos basado en esa clasificación.

A diferencia de K-means, que es un algoritmo no supervisado y donde la “K” significa la cantidad de “grupos” (clusters) que deseamos clasificar, en K-Nearest Neighbor la “K” significa la cantidad de “puntos vecinos” que tenemos en cuenta en las cercanías para clasificar los “n” grupos -que ya se conocen de antemano, pues es un algoritmo supervisado.

Su funcionamiento es el siguiente:

- 1.-Calcular la distancia entre el ítem a clasificar y el resto de ítems del dataset de entrenamiento.
- 2.-Seleccionar los “k” elementos más cercanos (con menor distancia, según la función que se use)
- 3.-Realizar una “votación de mayoría” entre los k puntos: los de una clase/etiqueta que <<dominen>> decidirán su clasificación final.

Teniendo en cuenta el punto 3, veremos que para decidir la clase de un punto es muy importante el valor de k, pues este terminará casi por definir a qué grupo pertenecerán los puntos, sobre todo en las “fronteras” entre grupos.

Las formas más populares de “medir la cercanía” entre puntos son la distancia Euclidiana o la Cosine Similarity (mide el ángulo de los vectores, cuanto menores, serán similares). Recordemos que este algoritmo funciona mejor con varias características de las que tomamos datos (las columnas de nuestro dataset).

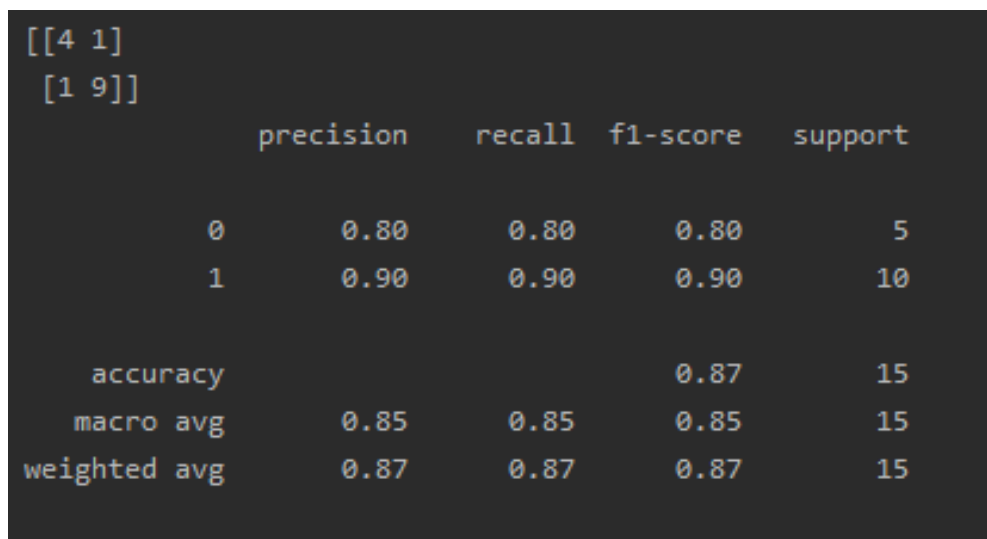
Entrenamiento usado:

Se utilizó la Table Testing. En el cual se usa una muestra de la tabla de datos para entrenar el modelo y los datos restantes se utilizan para probar el modelo.

Análisis de los modelos:

1: Red Neuronal (MLPClassifier) Perceptron multicapa clasificador.

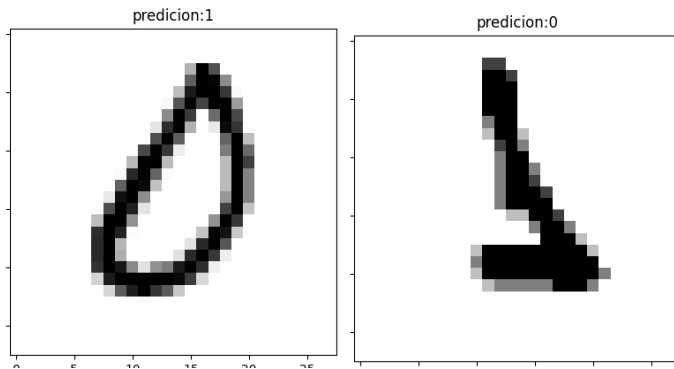
Se usó en la tabla testing en este caso un 50% para el testing lo cual son 15 datos, en el cual podemos confirmar, por la cantidad de datos probados que la clasificación es medianamente correcta, obteniendo los resultados siguientes:



```
[[4 1]
 [1 9]]
```

	precision	recall	f1-score	support
0	0.80	0.80	0.80	5
1	0.90	0.90	0.90	10
accuracy			0.87	15
macro avg	0.85	0.85	0.85	15
weighted avg	0.87	0.87	0.87	15

Como podemos observar el modelo nos dio una precisión del 87% la cual es una precisión medianamente buena y el error el cual es de 13% pues aún se mantiene pequeño, donde podemos ver que el modelo, calificar a un cero y un uno de una forma equívoca, pero aun así es un buen resultado.



las dos clasificaciones erróneas.

2: Árboles de decisión:

Igualmente para este modelos se usó un 50% de los datos para el testing, el cual tiene una mejora, más considerable a diferencia del anterior modelo, mostrando los siguientes resultados:

```
[[7 0]
 [1 7]]
```

	precision	recall	f1-score	support
0	0.88	1.00	0.93	7
1	1.00	0.88	0.93	8
accuracy			0.93	15
macro avg	0.94	0.94	0.93	15
weighted avg	0.94	0.93	0.93	15

Obtuvo un 6% más de precisión que el anterior modelo, el cual se traduce a un error del 7%, donde un 1 se clasificó erróneamente como un cerro, se considera que aun así la clasificación es bastante buena, incluso mejor que la anterior donde ahora solo existe un error, en vez de dos.

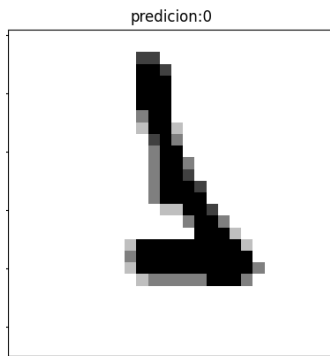


imagen erróneamente clasificada por el modelo de árboles de decisión.

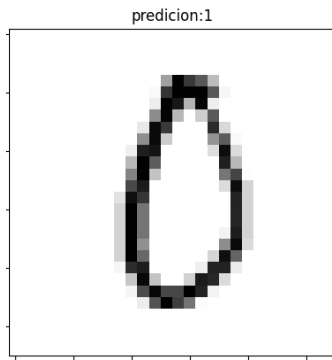
3 Knn:

Igualmente la tabla testing era del 50% de los datos, obteniendo así los siguientes resultados:

```
[[7 1]
 [0 7]]
```

	precision	recall	f1-score	support
0	1.00	0.88	0.93	8
1	0.88	1.00	0.93	7
accuracy			0.93	15
macro avg	0.94	0.94	0.93	15
weighted avg	0.94	0.93	0.93	15

observando, se obtuvo una precisión igual que el modelo anterior de 93% lo cual, inferimos que el error es de 7%, donde se clasificó un cero como un uno, por lo cual este algoritmo resulta igual de eficaz que el anteriormente mostrado.



el cero clasificado erróneamente.

Bueno los dos últimos algoritmos usados (knn y árboles de decisión) mostraron resultados de clasificación muy buenos, mientras la red neuronal MPLC logró una precisión menor del 90%, puede ser que estos resultados, se hayan reproducido por el tamaño del dataset, pero de forma general se obtuvieron muy buenas clasificaciones donde el promedio de error para todos los algoritmos utilizados fue de una clasificación equivocada, que en nuestra opinión son buenos resultados.

Tabla de comparación:

Algoritmos	Precisión global	Precisión positiva	Precisión negativa	Falsos positivos	Falsos negativos	Asertividad positiva	Asertividad negativa
MPLC	86.66%	90%	80%	20%	10%	90%	80%
Arb decisión	93.33%	87.5%	100%	0	12.5%	87.5%	87.5%
Knn	93.33%	100%	87.5%	12.5%	0	100%	100%

Conclusión:

Horacio: Bueno fue un proyecto muy complicado, pero amplió un poco más la clasificación de los algoritmos y en cuáles tipos de conjunto de datos es mejor usar cada uno, como se había visto anteriormente en el curso cada algoritmo funciona de una mejor manera dependiendo de nuestro conjunto de datos, aunque su realización fue un duro momento tanto físicamente como mentalmente.

Cardona: El comparar los algoritmos para casos específicos nos sirve bastante para saber cual usar posteriormente en casos similares. Ya que como vimos en clases, cada uno funciona de mejor forma para los tipos de datos y los resultados que se quieren obtener de estos.

Fuentes:

1.10. Decision Trees — scikit-learn 0.24.2 documentation. (s/f)., de Scikit-learn.org
website: <https://scikit-learn.org/stable/modules/tree.html>

A Beginner's Guide to Multilayer Perceptrons (MLP). (s/f)., de Pathmind.com
website: <https://wiki.pathmind.com/multilayer-perceptron>

Array.Shape() giving error tuple not callable. (s/f)., de Stackoverflow.com website:
<https://stackoverflow.com/questions/25125168/array-shape-giving-error-tuple-not-callable>

Dataset columns throwing KeyError. (s/f). R, de Stackexchange.com website:
<https://datascience.stackexchange.com/questions/54890/dataset-columns-throwing-keyerror>

Gallagher, J. (2020, julio 21). Python string to int() and int to string tutorial., de Careerkarma.com website: <https://careerkarma.com/blog/python-string-to-int/>

How to interpret scikit's learn confusion matrix and classification report? (s/f)., de Stackoverflow.com website:
<https://stackoverflow.com/questions/30746460/how-to-interpret-scikits-learn-confusion-matrix-and-classification-report>

KeyError in the below attached code. (2018, junio 1)., de Analyticsvidhya.com website:

<https://discuss.analyticsvidhya.com/t/keyerror-in-the-below-attached-code/64774/6>

Nair, A. (2019, junio 20). A beginner's guide to Scikit-learn's MLPClassifier., de Analyticsindiamag.com website:

<https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/>

Python Conditions. (s/f). R, de W3schools.com website:
https://www.w3schools.com/python/python_conditions.asp

Sanjay, M. (2018, octubre 26). MachineLearning — KNN using scikit-learn. R, de Towards Data Science website:
<https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75>

Wikipedia contributors. (2021, marzo 23). Multilayer perceptron., de Wikipedia, The Free Encyclopedia website:
https://en.wikipedia.org/w/index.php?title=Multilayer_perceptron&oldid=1013769694

Juan Ignacio. (Julio 10, 2018). Clasificar con K-Nearest-Neighbor., de Aprende Machine Learning Sitio web:
<https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/>

