Entrenamiento de Modelos de Clasificación sobre el Conjunto de Datos MNIST-C (Versión 'Fog')

Evaluación: Trabajo Práctico 02

Materia: Laboratorio de datos - FCEyN - UBA

Grupo : “Seguidores de Navathe”

Autor : Kamala Jimeno Leiton, Francisco Peix y Carolina Cuiña

Fecha : 25/02/2025 - 10/03/2025

Introducción

En el presente análisis trabajaremos con el dataset MNIST- C en su versión “Fog”, en donde cada fila que este posee hace referencia a una imagen, que representa un dígito entre el 0 y el 9, escrito a mano. Cada figura se compone de diversos atributos, y cada uno describe a un píxel de ésta; además de una columna adicional que indica el dígito que hace referencia a la imagen representada en esa fila.

El valor de la celda hace referencia a un color determinado, que se encuentra en una escala de grises. El 0 es el negro y, a medida que aumenta su valor, el tono es más claro, hasta el 255, que es el blanco.

Nuestro objetivo es encontrar patrones en los píxeles para identificar aquellos más relevantes para la clasificación de los dígitos. Posteriormente, aplicaremos modelos de clasificación, tanto en un problema de clasificación binaria (dígitos 0 y 1) como en la clasificación multiclase de los 10 dígitos, explorando diferentes enfoques y técnicas para optimizar su desempeño.

Es esperable que sea más complicado distinguir algunos pares de dígitos que otros. Por ejemplo, es más sencillo diferenciar al 0 del 1 que al 0 del 9.; sin embargo, hay dígitos que visualmente son muy similares, como el 8 y el 3. En resumen, en los casos donde los dígitos presentan semejanzas, la tarea de diferenciarlos será más difícil, mientras que cuando las formas son claramente distintas, la clasificación resultará más sencilla.

A lo largo del análisis, evaluaremos la efectividad de distintos atributos y modelos mediante métricas como la exactitud (*Accuracy Score*), utilizando técnicas como la normalización, selección de atributos y validación cruzada para mejorar la capacidad predictiva de los algoritmos implementados.

Para la selección de modelos de clasificación, recurriremos a utilizar dos tipos de ellos:

- En primer lugar, para clasificación binaria, implementaremos modelos basados en KNN, donde utilizaremos distintos conjuntos de atributos en las que las clases puedan diferir (0 y 1); y evaluaremos la *performance* obtenida en cada uno de ellos, para luego poder compararlas y poder informar qué conjunto de pixeles fue el más útil para poder determinar si una imagen corresponde a una clase o la otra.
- Luego, para la clasificación multiclase, diseñaremos árboles de decisión; y también, a través de la validación cruzada, veremos qué profundidad será la más óptima para elegir, la cual se utilizará para analizar qué tan bien se predicen los dígitos en las imágenes.

A lo largo del informe, presentaremos distintos tipos de gráficos que facilitarán la interpretación de los datos, proporcionando una mejor comprensión en comparación con la observación directa de los datos crudos. Estos gráficos también permitirán comparar las métricas obtenidas y seleccionar el modelo más eficaz. Finalmente, en el último apartado, implementaremos una matriz de confusión para analizar el desempeño del modelo en la predicción de cada dígito

Análisis Exploratorio del Dataset original

En este trabajo, utilizaremos el conjunto de datos de imágenes MNIST-C en su versión “Fog”. Este dataset contiene 70.000 instancias (filas), donde cada una representa una imagen de un dígito escrito a mano, comprendido entre el 0 y el 9. Los atributos están distribuidos en 786 columnas: la primera corresponde a un índice que solo indica la cantidad de imágenes presentes en la fuente, por lo que no aporta información relevante para el análisis y no será considerada; las siguientes 784 columnas representan los valores de los píxeles que describen cada imagen, mientras que la última columna, denominada ‘labels’, indica el dígito asociado a cada instancia. La variable de interés tiene 10 clases, correspondientes a los dígitos del 0 al 9, todas ellas se representan en una gran cantidad a lo largo del conjunto de datos.

Los píxeles mencionados se presentan con un número que corresponde a un tono determinado dentro de una escala de grises, siendo 0 el color negro y 255 blanco; y resulta importante mencionar que cada imagen posee dicha cantidad de píxeles debido a que estas se presentan con un tamaño de 28x28 (imagenes de 8-bits). Además, decidimos observar si la cantidad de clases se encuentra balanceada con el objetivo de profundizar nuestro análisis y observar en detalle con lo que estamos trabajando. Lo examinado se puede encontrar en la sección **Anexo**[1].

Al observar el dataset original, notamos que los valores de cada fila se distribuían en distintos intervalos, por lo que decidimos normalizar el conjunto de datos antes de realizar cualquier tipo de análisis. Esta normalización permite resaltar mejor las diferencias concretas al transformar el dominio de los valores a un rango entre 0 y 1, donde 0 representa el color negro y 1 el color blanco.

Para analizar la existencia de atributos más útiles que otros al momento de predecir el número correspondiente a la imagen de cada fila, decidimos calcular las medias de cada clase y, una vez obtenidas, graficar la imagen resultante con el objetivo de detectar cuales son los atributos que nos interesan para poder diferenciar el dígito. Una vez hecho esto, consideramos que los píxeles más relevantes de la imagen serán quienes se encuentren en un tono más claro que el resto. Para poder reflejar cuales son, decidimos encuadrarlos.

Figura 1

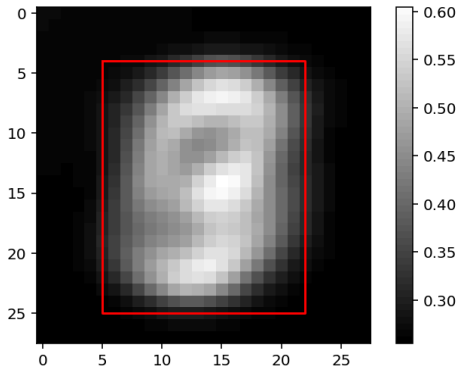


Figura 2

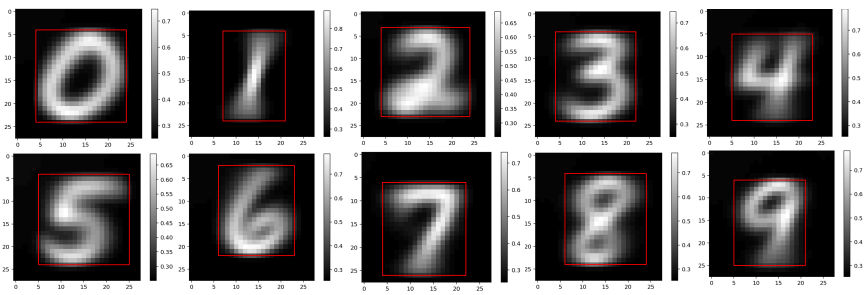


Figura 1. *Gráfico que muestra el promedio de todo el dataframe, con los píxeles cuyo valor es mayor a 0.3 encuadrados*

Figura 2. *Gráficos que muestran los promedios de cada clase, con los píxeles cuyo valor es mayor a 0.3 encuadrados .*

Los píxeles que se encuentran encuadrados en el gráfico del lado izquierdo (ver **Figura 1**) se refieren a aquellos atributos que pueden ser útiles al momento de identificar el dígito presentado en la imagen, pues son aquellos que suelen ser claros (su valor normalizado es mayor a 0.3). En el lado derecho, se encuentra representado, dentro cada encuadre, los conjuntos de atributos considerados importantes para identificar a cada clase por separado (ver **Figura 2**). Podemos observar que los atributos encuadrados son similares en todos los casos y, además, estos se ubican en el centro de la imagen. Así, podríamos asegurar que aquellos atributos más relevantes para realizar la tarea establecida son aquellos píxeles localizados en el centro de la imagen; mientras que los menos destacables son los que se encuentran en los bordes de ésta. Estos últimos podrían ser descartados, pues no brindan ninguna información notable para poder cumplir con el objetivo principal de determinar la clase correspondiente a la imagen. De hecho, como respuesta a lo observado decidimos utilizar la información recolectada para realizar los modelos de predicción posteriores.

Otra particularidad que se obtuvo a partir de estos gráficos es que, si miramos la escala de los colores que representa cada píxel, podemos notar que éstos suelen encontrarse entre un mismo rango de valores, entre los 0.25 y 0.75 aproximadamente (con la fuente de datos normalizada). Esto se debe a que el conjunto de datos de imágenes está en una versión especial (versión “Fog”), por lo que no se suelen presentar valores tan abruptos (0 o 1), sino que las tonalidades de colores suelen ser similares.

En cuanto a las semejanzas entre ciertos dígitos, es evidente que algunos presentan una mayor similitud entre sí. Por ejemplo, las imágenes correspondientes al dígito 3 se diferencian con mayor facilidad de las del dígito 1, mientras que resulta más complejo distinguir entre las representaciones del dígito 8 y las del dígito 3. Esto se debe a que la forma del dígito 8 es muy similar a la del 3, mientras que la del 3 y la del 1 presentan diferencias más marcadas y claramente distinguibles.

Para justificar nuestra observación, utilizando las formas promedio de cada dígito graficadas previamente, decidimos realizar nuevos gráficos en donde representamos las distinciones que tienen los números a comparar (en nuestro caso, entre el 3 y el 1; y 8 y el 3, respectivamente). Para lograr esto, aplicamos la resta entre los conjuntos anteriores para poder ver en cada píxel cuál número predomina más:

Figura 3 **Figura 4**

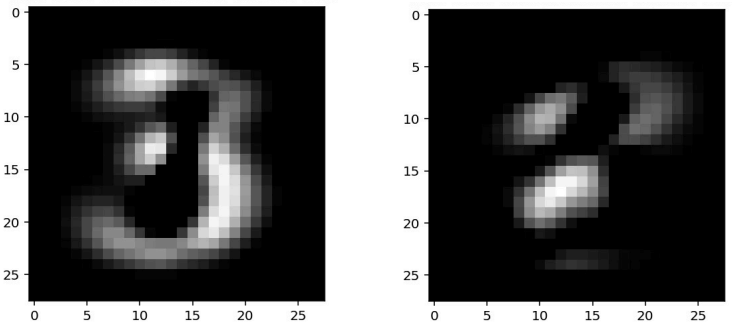


Figura 3. *Gráfico que muestra la diferencia entre las formas de los números 3 y 1*
Figura 4. *Gráfico que muestra la diferencia entre las formas de los números 8 y 3*

Por un lado, se puede observar, en la **Figura 3**, que el número 3 es legible a simple vista, ya que los píxeles que forman al número 1, en su mayoría, se encuentran en otra posición a la del número 3. Por otro lado, el gráfico derecho (ver **Figura 4**) muestra una escena diferente, donde el 8 y el 3 no se distinguen fácilmente, ya que muchos de los píxeles que pertenecen a sus figuras son compartidos, haciendo difícil distinguirlos. Este problema a la hora de identificar los números puede ser una causa del decrecimiento de la exactitud de los modelos de clasificación al trabajar con ellos.

Una vez analizadas las semejanzas entre distintos dígitos, buscamos enfocarnos en encontrar parecidos entre distintas imágenes de un mismo número. Para ello, tomamos como ejemplo la clase de el número ‘0’.

Figura 5

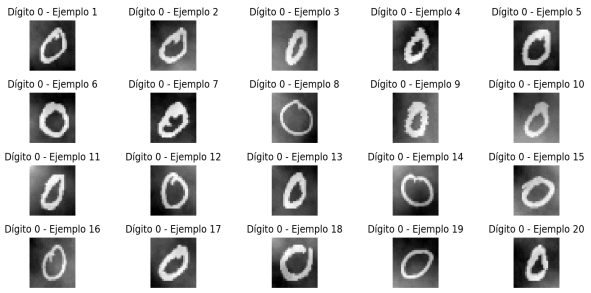


Figura 6

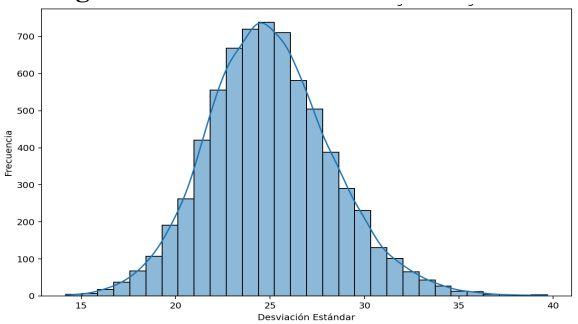


Figura 5. *Muestras de imágenes correspondientes al dígito 0* .
Figura 6. *Distribución de la desviación estándar de las imágenes del dígito 0*

Notamos, al comparar muchos ejemplares de éstos, que no siempre sus píxeles representativos (claros) suelen estar ubicados en el mismo lugar, pero que aún así son identificables con el mismo dígito. Para tener una mejor comparación, decidimos representar un histograma con la distribución de la desviación estándar que tenían las imágenes correspondientes a dicha clase. En él, pudimos observar que la desviación estándar más usual que las imágenes correspondientes presentan suele estar en el 25 aproximadamente, con una frecuencia de casi 700 figuras.

Para ampliar nuestro análisis, también realizamos los mismos pasos, pero esta vez con la clase correspondiente al número ‘5’.

Figura 7

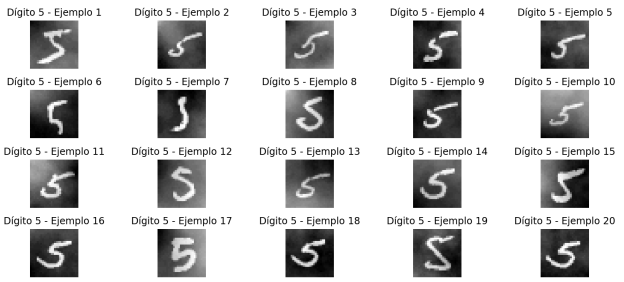


Figura 8

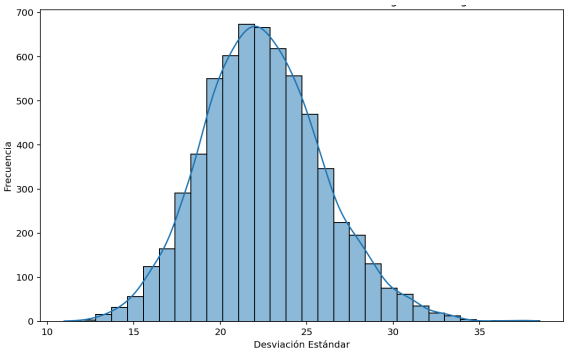


Figura 5. *Muestras de imágenes correspondientes al dígito 5* .
Figura 6. *Distribución de la desviación estándar de las imágenes del dígito 5*

En esta muestra (ver **Figura 8**), se puede notar que los datos nuevamente poseen unos valores de desviación estándar similares a los de la anterior clase, en donde se observa que la desviación estándar más común suele ser entre los valores 20 y 25, con una frecuencia de casi 700 imágenes (si tenemos una desviación estándar de 20, significa que, en promedio, los valores de los píxeles de las imágenes del dígito 0 se desvían 20 unidades de la media. En otras palabras, hay una variabilidad considerable en los valores de los píxeles entre las diferentes imágenes del dígito 0 y 5).

En cuanto a la dificultad en la exploración de los datos, podemos observar una diferencia entre el dataset trabajado y los utilizados previamente en clases, como los de Titanic o Iris, debido a sus respectivas composiciones. No obstante, esta distinción no representa un obstáculo significativo para la exploración, ya que conocemos la estructura del conjunto de datos: cada atributo corresponde a un píxel específico de la imagen (784 columnas con valores numéricos en escala de grises, representando imágenes de 28x28 píxeles). Estos atributos serán nuestras variables explicativas (X), mientras que la columna 'labels' indica el dígito asociado a cada imagen, comprendido entre 0 y 9 (10 clases en total), y representará nuestra variable a explicar (Y). Con esta información, podremos entrenar modelos de la misma forma en que lo hemos aprendido, lo que nos permite realizar el análisis de manera eficiente y precisa.

Clasificación binaria

Para esta parte del análisis, fue necesario construir un nuevo DataFrame a partir de la fuente de datos original, que contuviera únicamente el subconjunto de imágenes correspondientes a los dígitos 0 y 1. Este subconjunto está compuesto por 14.780 muestras, lo que representa aproximadamente un 21% de todas las muestras. Observamos que el subconjunto no está balanceado respecto a los dos dígitos a predecir, ya que existe una diferencia notable de 1.000 muestras entre la cantidad de imágenes correspondientes a los dígitos 0 y 1. Para visualizar esto con mayor claridad, realizamos un gráfico de barras:

Figura 9

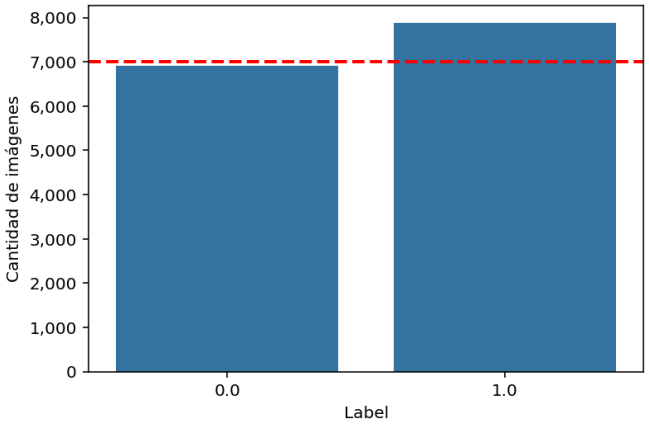


Figura 9. *Gráfico de barras que muestra la cantidad de veces que aparece cada dígito*

Previo a separar los datos en conjuntos de *Train* y *Test*, decidimos recortar las columnas cuyos píxeles no eran relevantes a la hora de distinguir el dígito en cuestión (analizado en el apartado anterior). Luego, definimos las variables X (*variables explicativas*), que corresponden a las columnas del archivo, sin incluir aquella denominada *'labels'* (píxeles de la imagen en cuestión); y la variable Y (*variable de respuesta*), en la que sí se encuentra el atributo *'labels'* (número al que corresponde la imagen). Elegimos un tamaño de *Test* de 0.2 (un 20% del total de datos presentes), asegurándonos la consistencia de la división de estos datos utilizando *random_state*[1].

Una vez realizada la división de los casos, ajustamos un modelo de KNN con los datos de *Train* donde consideramos distintos conjuntos de pocos atributos. Para elegirlos, analizamos los gráficos que muestran los promedios de los valores que poseían los atributos en cada clase realizados anteriormente. Observamos que los atributos cuyos píxeles se encuentran en el centro de la imagen suelen diferenciar las imágenes entre 0 y 1, pues en el 0 se hallan valores menores a 0.45; mientras que, en el 1, estos datos suelen ser mayores. Esto hace que, en los píxeles centrales del 0, se contenga frecuentemente colores más oscuros a comparación del otro dígito. Teniendo en cuenta esto, consideramos 2 conjuntos de 3 atributos cada uno: "centro vertical" ([434, 406, 462]) y "centro horizontal" ([412, 413, 414]). Además, notamos que hay zonas que no necesariamente pertenecen al

[1] Parámetro de número pseudo-random que nos permite reproducir el mismo *train test split* cada vez que es ejecutado el código

centro de la imagen, donde si el valor del píxel es mayor a 0.45 en uno de los dígitos, esto no se cumple en el otro. Para este caso, seleccionamos el conjunto “esta en el 0 y falta en el 1” ([262, 407, 468]), el cual contiene uno de los *Accuracy Scores* más altos entre los conjuntos elegidos. Luego, probamos aumentando la cantidad de atributos a considerar, probando con 5 y 10 atributos, llamados “5 atributos” ([181, 263, 350, 513, 659]) y “10 atributos” ([127, 184, 269, 323, 328, 384, 428, 434, 597, 634]).

Luego de seleccionar los conjuntos, ajustamos un modelo de KNN en los datos de *Train*, considerando 5 vecinos como base para poder comparar los *Accuracy Scores* de los distintos conjuntos. Para poder visualizar la diferencia, realizamos una tabla que muestra, para cada conjunto de atributos, el puntaje de *Train* y *Test* obtenido:

Tabla 1.

Conjunto	Train	Test
centro vertical	0.957459	0.956022
centro horizontal	0.950440	0.946888
esta en el 0 y falta en el 1	0.992304	0.992219
5 atributos	0.892422	0.845737
10 atributos	0.990020	0.986806

Aquí (**Tabla 1)** podemos observar que, siguiendo nuestros criterios de elección de los conjuntos, la exactitud de nuestro modelo es muy alta, pues todos los puntajes se encuentran arriba de 0.84. Se evidencia que los conjuntos con mayor puntaje son ‘esta en el 0 y falta en el 1’ y ‘10 atributos’. Esto podría deberse a que, en el caso de ‘esta en el 0 y falta en el 1’, aunque sean pocos atributos, éstos son claves ya que en ellos hay mucha distinción entre la imagen promedio del 0 y del 1, y los píxeles elegidos están distanciados entre sí, mientras que ‘10 atributos’ tiene más categorías para poder comparar y tiene más posibilidades de acertar en el dígito en cuestión. Por otro lado, los que están concentrados en el centro de la imagen pueden contener errores, ya que al estar enfocados en un punto, si la imagen de uno de los números se encuentra corrida (no centrada) y coinciden los valores, el modelo debe elegir el dígito al azar, lo que deja margen de error, provocando falta de exactitud a el modelo. El conjunto que tiene el puntaje más bajo es ‘5 atributos’. En este caso, consideramos también píxeles que estaban en los bordes de las figuras a comparar, lo que hace que suceda una situación similar al caso de los píxeles centrales, donde, si ambas figuras tienen valores parecidos, el entrenamiento del modelo puede volverse impreciso. Así, podemos concluir que la exactitud del modelo depende no solo de la cantidad de atributos que uno elija, sino también de que estos atributos elegidos sean claves para poder distinguir entre las clases que uno quiere clasificar.

Posteriormente, buscamos comparar estos mismos atributos usando distintas cantidades de k vecinos en búsqueda de la más óptima. Para eso, diseñamos un gráfico de líneas que muestra, a medida que aumenta k, qué sucede con el *Accuracy Score* de los modelos entrenados con los distintos conjuntos de atributos:

Figura 10

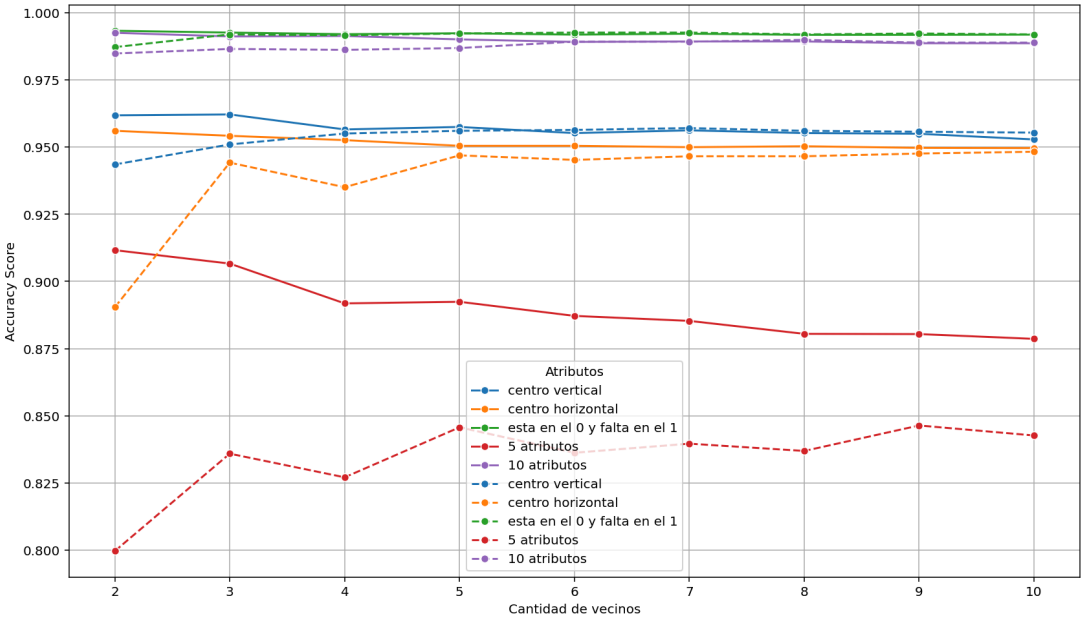


Figura 10. *Gráfico que muestra la relación entre la cantidad de vecinos y el accuracy score para cada conjunto de atributos*

En principio es importante señalar que la exactitud de nuestro modelo entrenado siempre se encuentra arriba de 0.8, lo que muestra un buen nivel de desempeño a la hora de distinguir entre dígitos para los conjuntos de *Train* (los que están representados con líneas contínuas) y de *Test* (dibujados con líneas punteadas) (ver **Figura 10**). Encontramos que 'esta en el 0 y falta en el 1' y '10 atributos' son los conjuntos más exactos, y que su puntaje se mantiene casi constante sin importar la cantidad de vecinos en los que se los evalúe. En contraste, observamos que '5 atributos' tiene el menor puntaje y , en el caso de su conjunto de *Train,* su exactitud decrece a medida que aumenta k. En general, el número elegido k = 5 puede ser conveniente a la hora de construir nuestro modelo pues evita el sobreajuste, pero también logra conseguir uno de los *Accuracy Scores* más altos en tanto el conjunto de *Test* como el de *Train*.

Clasificación Multiclase

En esta sección nos enfocaremos en responder, dada una imagen, la siguiente pregunta: ***¿A cuál de los 10 dígitos corresponde la imagen?***

Para ello, decidimos abarcar únicamente aquellos atributos que consideramos que contienen información que nos será de utilidad al momento de responder la pregunta. Éstos son los píxeles claros que fueron previamente marcados en el encuadre de la **Figura 1**.

Una vez aclarado esto, nos enfocaremos en crear modelos de clasificación (en este apartado usaremos Árboles de Decisión), con distintos hiperparámetros (profundidades), y quedarnos con aquel que haya obtenido la mejor *performance* al momento de clasificar las imágenes utilizadas para evaluarlos.

Para entrenar los modelos y evaluarlos, al igual que en la sección de clasificación binaria, definimos las variables X (*variables explicativas*), que corresponden a las columnas del archivo, sin incluir aquella denominada *'labels'* (píxeles de la imagen en cuestión); y la variable Y (*variable de respuesta*), en la que sí se encuentra el atributo *'labels'* (número al que corresponde la imagen). Separamos los datos, destinando un 75% para el conjunto de desarrollo y el 25% restante para la validación, donde este último será utilizado solamente para el final del proceso de selección.

A continuación, decidimos ajustar modelos de árboles de decisión utilizando únicamente el conjunto de desarrollo, probando con distintas profundidades, encontradas entre 1 y 10. Para medir su *performance*, utilizamos la exactitud (*Accuracy Score*).

Figura 11

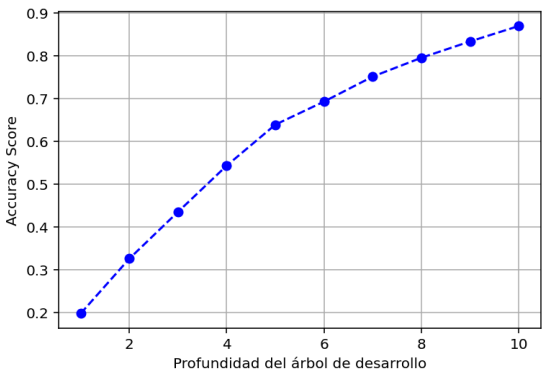


Figura 11. *Performance del árbol de decisión según profundidad, utilizando el conjunto de desarrollo*

En este gráfico (ver **Figura 11**) se puede ver que, a medida que se incrementa la profundidad del árbol, el valor del *Accuracy Score* también aumenta. En un comienzo, se decidió elegir la profundidad 10 como la óptima, ya que era la que resultaba tener la métrica más alta. Pero, si se analiza detenidamente, a partir de la altura 5 en adelante, la métrica crece en menor cantidad, teniendo cambios cada vez menos significativos. Lo que puede ser un indicio de sobreajuste.

A partir de esto aplicaremos el método de validación cruzada para comparar y seleccionar distintos árboles de decisión, utilizando los mismos hiperparámetros que en la anterior ocasión para poder determinar cuál es la altura adecuada. En este punto, fue importante establecer la *cantidad de folds*[2] a utilizar; donde elegimos tener 8 particiones.

Figura 12

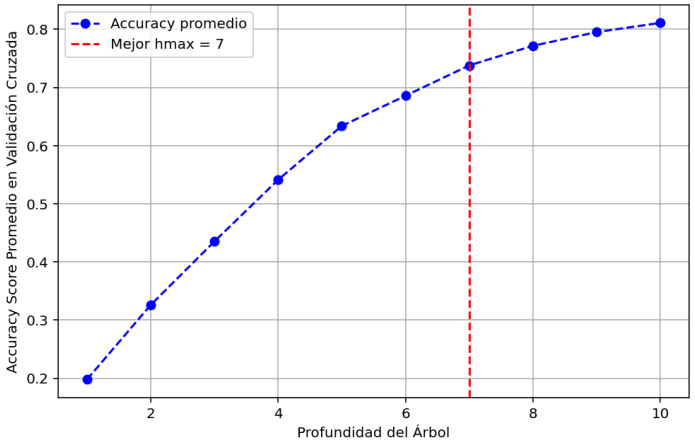


Figura 12. *Exactitud promedio en validación cruzada del árbol de decisión según profundidad*

En este gráfico (**Figura 12**), se observa como incrementa la exactitud (A*ccuracy Score*) promedio obtenida por cada hiperparámetro en el conjunto de desarrollo mediante la implementación del algoritmo de *K-folding* con respecto a la profundidad del árbol.

Se puede observar que, a medida que aumenta la profundidad del árbol, incrementa el *Accuracy Score* promedio; pero, a partir de la altura 5, lo hace cada vez de manera menos significativa. Notamos también que, luego de hacer validación cruzada, la *performance* del árbol da resultados más bajos que aquellos encontrados en el gráfico anterior (**Figura 11**). Esto se debe a que, mediante el método de *K-folding*, se divide el conjunto en k subconjuntos, entrenando y evaluando modelo k veces con los mismos hiperparámetros, de manera que cada parte actúa una vez como conjunto de prueba, mientras que las partes restantes se usan para el entrenamiento. Esto genera distintos parámetros internos en el modelo resultante. Finalmente, la performance se obtiene promediando el rendimiento de cada iteración, lo que permite que la métrica obtenida considere una mayor variedad de instancias de entrenamiento y prueba dentro del conjunto de desarrollo.

Esto sucede ya que aquí estamos evaluando no solo el conjunto de desarrollo, sino que el modelo también utiliza el conjunto de evaluación o *held-out.* Esto se puede apreciar en el cambio de valores en los que se encuentra el eje Y. Para analizar los datos obtenidos con mayor detención, los invitamos a visitar la sección **Anexo**[2] donde se encuentra una tabla que presenta la diferencia entre el puntaje obtenido en el árbol de desarrollo y el de validación cruzada, en las distintas profundidades.

[2] Cantidad de particiones del conjunto, del mismo tamaño.

Aunque el valor máximo del *Accuracy Score* es cuando el hiperparámetro es igual a 10, siguiendo el Principio de la Navaja de Ockham que establece que cuando existen múltiples modelos con un desempeño de predicción similar, es recomendable elegir el más simple, finalmente decidimos elegir como más óptimo aquel de altura 7. Disminuyendo así el riesgo de que ocurra sobreajuste o la pérdida de explicatividad del modelo (conservando una exactitud relativamente alta).

Finalmente, una vez elegido el mejor modelo para poder satisfacer nuestro objetivo, lo entrenamos utilizando el conjunto de desarrollo (*dev*); y lo evaluamos en el conjunto de validación (*held-out*). A continuación presentamos los resultados obtenidos:

Tabla 2.

Conjunto	**Accuracy score promedio**
Conjunto de Desarrollo (dev)	0.751848
Conjunto de Evaluación (held-out)	0.734000

Aquí (**Tabla 2**) se puede observar que el conjunto de desarrollo (*dev*) tiene una exactitud mayor a los resultados obtenidos en el conjunto de evaluación (*held-out*). Esto se puede presentar por la similitud entre ciertos dígitos pues, como fue marcado anteriormente, muchos de ellos comparten valores similares en ciertos atributos.

Para poder observar con mayor detenimiento el motivo del valor de *performance* hallado en el conjunto de validación, decidimos realizar una matriz de confusión, con el fin de lograr evaluar los distintos tipos de errores que se presentaron para las clases en cuestión:

Figura 13

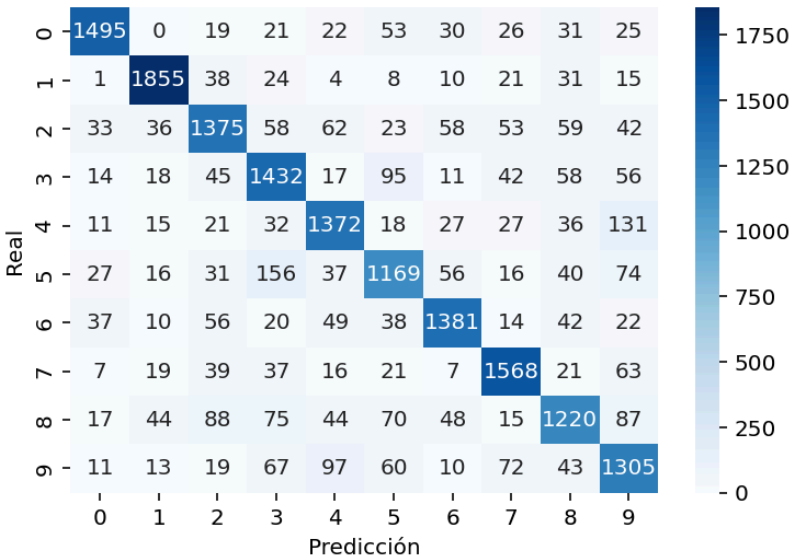


Figura 13. Matriz de confusión

Utilizando los datos provistos por la imagen (ver **Figura 13**), calculamos los porcentajes de predicciones erróneas y acertadas, las cuales se pueden apreciar en detalle en la sección de **Anexo** [3]. Observamos que los 3 dígitos con mejor desempeño, es decir mayor porcentaje de predicciones correctas, fueron el 1 con un 92,42% de aciertos, el 7 con 87,2% y por último el 0 con 86,82%.

Por otro lado, los dígitos con menor exactitud en la predicción fueron el 8, con un 28,57% de predicciones erróneas; el 5, con un 27,9%; y el 9, con un 23,1%.

Analizando el caso del número 8, notamos que los dígitos con los que más se suele confundir son el 2, 9 y 3. Esto es coherente ya que podemos notar el parecido entre la forma del 8 con la del 3 y 9. En particular, el 8 es clasificado erróneamente como un 2 o un 9 en el 18% de los casos cada uno, mientras que el 3 representa un 15% de las confusiones.

Observamos que el 5 es identificado erróneamente como un 3 con gran frecuencia. De hecho, viendo las predicciones de este dígito, casi un 34% del total de los casos donde está mal clasificado corresponden al número 3.

Con respecto al dígito 9, cuando se lo predice erróneamente, generalmente da como resultado el 4. Este representa el 25% de los errores que este posee.

Además, notamos que el único caso donde la celda vale 0 (situación ideal en una matriz de confusión porque significa que no hay predicciones erróneas) es cuando el modelo no se confunde nunca al 0 con el número 1.

Conclusión

Luego de haber realizado este vasto informe, hemos llegado a una serie de conclusiones que nos parecen importantes reportar:

- Encontramos que aquellos atributos más relevantes para realizar la tarea establecida son aquellos píxeles localizados en el centro de la imagen, mientras que los menos destacables son los que se encuentran en los bordes de ésta.

- El parecido entre números puede ser una causa del decrecimiento de la exactitud de los modelos de clasificación, ya que muchos de los píxeles que pertenecen a sus figuras son compartidos, haciendo difícil distinguirlos.

- La exactitud del modelo depende no solo de la cantidad de atributos seleccionados, sino también de que estos sean relevantes para distinguir entre las clases a clasificar. No solo importa su cantidad, sino también su calidad. Por ejemplo, al entrenar los modelos, decidimos utilizar los píxeles que eran relevantes para la distinción de los dígitos, evitando aquellos que no aportaban información o que aumentaban la confusión.

- Al elegir los hiperparámetros adecuados para el modelo con el que estamos trabajando, es importante que éste evite el sobreajuste y logre una alta exactitud. Por ejemplo, la cantidad de vecinos elegida en el caso de KNN (en nuestro caso, elegimos 5 vecinos).

- La profundidad óptima de un árbol no siempre garantiza la mejor métrica utilizada para la comparación. Siguiendo el Principio de la Navaja de Ockham, en algunos casos es preferible optar por un modelo con una métrica ligeramente peor pero más sencillo, en lugar de uno más complejo con una métrica apenas superior, siempre que la diferencia no sea significativa. Este fue nuestro caso, en el que elegimos una profundidad de 7 en lugar de 10.

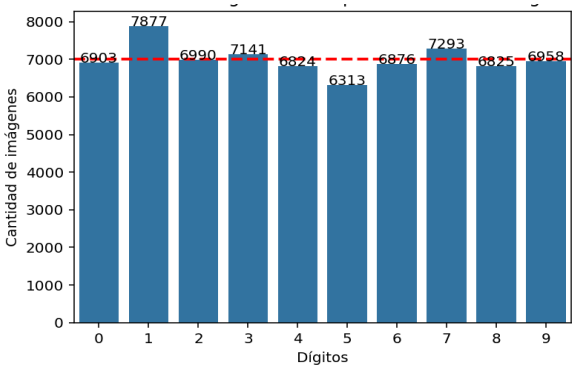
- Los dígitos con que tuvieron mejor desempeño son los que se diferencian más del resto por ejemplo el 1 es muy distinto en comparación al resto de los dígitos. Mientras que los que tienen siluetas más parecidas son más difíciles de confundir como por ejemplo el 8 con el 3.

Finalmente, tras la elaboración de este informe, reconocemos la importancia de no aferrarse a un único modelo para realizar la tarea, sino de construir y evaluar diversos modelos con distintos hiperparámetros. Este enfoque permite comparar su desempeño y seleccionar el más adecuado para alcanzar la meta propuesta, lo que se considera una buena práctica en el desarrollo de modelos de clasificación. Esta metodología no solo fue aplicada en este trabajo, sino que nos comprometemos a adoptarla en aquellos por venir.

Anexo

[1]

Un detalle no menor que se toma en consideración es, por ejemplo, analizar si la cantidad de clases se encuentra balanceada, esto es, que la cantidad de imágenes correspondiente a cada dígito (cantidad de filas que tienen mismo 'label') sea equitativa. Para poder analizar esto, realizamos un gráfico de barras, en donde se representa la cantidad de apariciones de cada label.



Cantidad de imágenes correspondientes a cada dígito

Aquí se puede observar que la cantidad de imágenes que cada clase (dígito) posee no es equitativa, por lo que estas no se encuentran balanceadas. Además, se puede apreciar que en algunos casos hay un gran contraste con la proporción que éstas tienen, como por ejemplo, la diferencia entre la clase correspondiente a el dígito '1' y la del '5'. Para facilitar la comprensión, trazamos una línea de corte en la cantidad 7000, para poder notar que únicamente los que pasan a esta son los dígitos '1', '3' y '7'; y además se puede apreciar el número que cada categoría posee.

[2]

Profundidad	scores promedio	Diferencia entre el score promedio previo
1	0.1979	
2	0.3260	+ 0.1281
3	0.4351	+ 0.1091
4	0.5412	+ 0.1061
5	0.6338	+ 0.0926
6	0.6856	+ 0.0518
7	0.7377	+ 0.0521
8	0.7717	+ 0.034
9	0.7950	+ 0.0233
10	0.8106	+ 0.0156

En esta tabla se mide el score promedio calculado en validación cruzada con respecto a la profundidad. Se construyó con el objetivo de mostrar que a medida que aumenta la

profundidad el cambio de score promedio es cada vez más leve. En especial en el rango [5,6] y más leve aún en el rango [8,10].
[3]

Dígito	Porcentaje de casos con erróneas predicciones	Porcentaje de casos con correctas predicciones	Número con el que más se lo confunde
0	13.18	86.82	5
1	7.58	92.42	2
2	23.57	76.43	8
3	19.91	80.09	5
4	18.82	81.18	9
5	27.9	72.1	3
6	17.25	82.75	2
7	12.8	87.2	9
8	28.57	71.43	2
9	23.1	76.9	4

Esta tabla se hizo en base a los datos de la matriz de confusión con el objetivo de poder comparar las predicciones por dígito. Se decidió plasmar los datos en porcentajes porque como no hay la misma cantidad de imágenes por dígito iba a ser muy complicado comparar los valores. Aquí se puede ver el porcentaje de predicciones erróneas y de predicciones correctas como también el número con el que más se confunde al dígito estudiado por ejemplo al 7 se lo confunde más con el 9. Además se resaltaron los dígitos con mejor desempeño y los que tuvieron peor desempeño.