Trabajo Práctico 02

Laboratorio de Datos
Verano 2025

Docentes:
Profesor - Pablo Turjanski
Profesora - Manuela Cerdeiro
Ayudante de 2da - Mateo Guerrero Schmidt

Equipo: “Corgi”
Integrantes:

- Máximo Mele
- Diego Horacio Hermida
- Juan Ignacio Bianchini

1. Introducción

Para este trabajo se hizo uso del conocido dataset “MNIST” en su versión corrompida “fog” con la intención de evaluar las capacidades de los modelos de clasificación KNN y DecisionTreeClassifier.

Primero se realizó un análisis exploratorio de los datos, buscando entender cómo estaban almacenados los datos, y se encontró que cada columna representa la intensidad de un pixel de la imagen, con valores entre 0 y 255, y la última columna “label” es la etiqueta que nos dice a cual digito se refiere esa imagen. Son 784 columnas, que permite rearmar una imagen cuadrada de 28x28.

Para el primer ejercicio, se utilizó el modelo de clasificación KNN buscando hacer una clasificación binaria entre aquellas imágenes que representan o un 0 o un 1. Se probaron diferentes conjuntos de atributos mediante algunos criterios cuantitativos elaborados por el equipo. Se pudo observar que con muy pocos atributos (Incluso con 3) se podía conseguir una alta exactitud en el modelo, siendo capaz de alcanzar un 98.5% o más. Se mantuvo siempre el mismo set de train y test, tal y como fue especificado en el enunciado.

Para el segundo ejercicio se utilizó el modelo de clasificación Decision Tree Classifier, en este caso para clasificación multiclase. En este ejercicio se utilizaron las técnicas de K-Folding para prevenir un sobreajuste del modelo. Se evaluaron diferentes profundidades del árbol buscando los mejores valores para la clasificación. También se probó variando el parámetro max_features, recortando las bordes (dejando una imagen de 18x18) y se analizó el impacto de los parámetros min_samples_split y min_samples_leaf. Una vez encontrados los mejores parámetros (spoiler: sólo importó el max_depth), se entrenó un modelo con exactitud superior al 81%.

2. Análisis Exploratorio

a. Los atributos son en este caso pixeles. Consideramos que los atributos que parecen ser más relevantes son aquellos que tienen un trazo de un dígito, mientras que aquellos que en general no son dibujados, como por ejemplo los bordes de las imágenes, serían menos relevantes para el análisis. Creemos que de mínima podemos eliminar los bordes, pero debe haber píxeles más relevantes para la detección que otros, por lo que quizás podemos encontrar algunas columnas en particular que nos permitan tener un buen desempeño a la hora de entrenar un modelo.

Para encontrar píxeles útiles a la hora de distinguir entre dígitos, se comenzó calculando las medianas de los valores de los píxeles para cada dígito del dataset. Las imágenes producidas (Figura 1) muestran claramente los dígitos a los que corresponden.

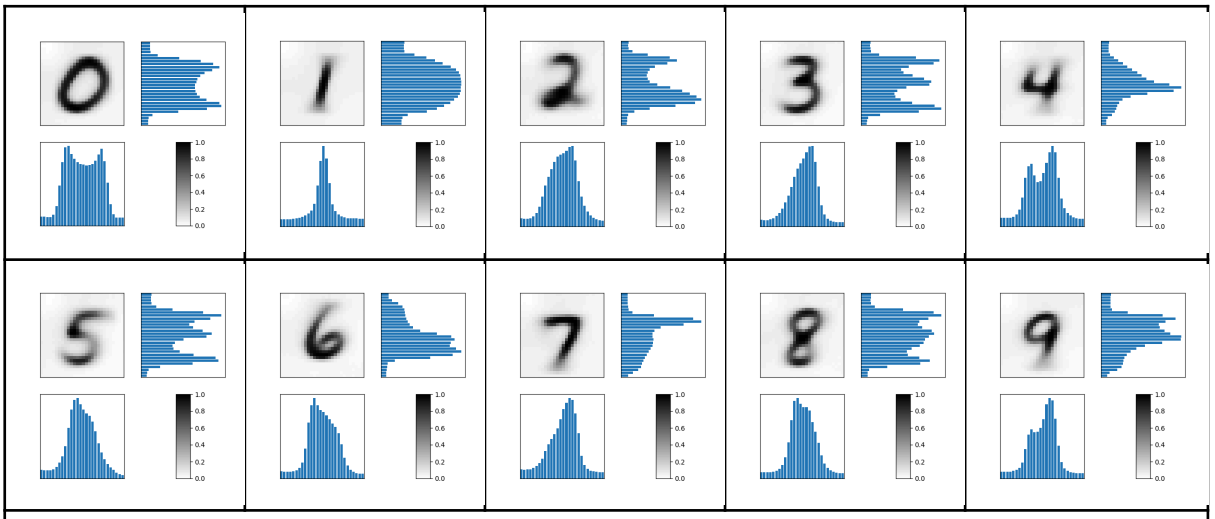

Figura 1. Mediana calculada para cada dígito, acompañada de la distribución vertical y horizontal de sus valores (suma de los valores de cada fila o columna respectivamente).

Más adelante, en la sección de clasificación binaria, se desarrollará cómo se utilizaron las medianas de los dígitos para encontrar los píxeles que mejor discriminan entre ceros y unos.

b. Al visualizar las medianas de los dígitos del dataset, encontramos ciertas similitudes entre algunos de ellos. Hay números que comparten trazos verticales u horizontales, que podrían hacer más difícil la tarea de diferenciar entre ellos. Ejemplos de esto son el 3, 5 y el 8 (Figura 2), o sino el 4 y el 9.

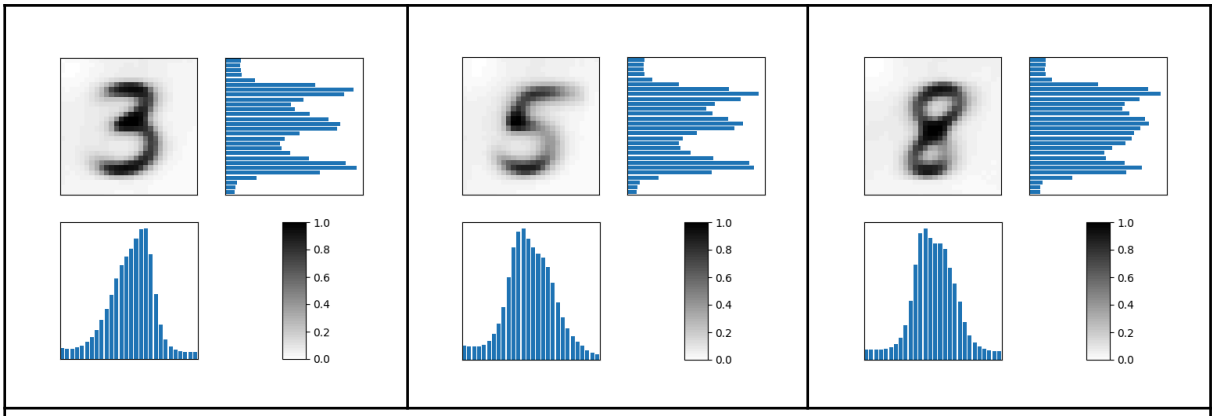

Figura 2. Gráficas del 3, 5 y 8 según la Figura 1. Visualmente el 3 y el 5 son muy similares en la curva inferior tanto como la superior, y cuando vemos las distribuciones generales de los tres números, son bastante parecidas.

En la matriz de confusión del modelo multiclase entrenado más adelante (Figura), se puede ver cómo estos dígitos son comúnmente confundidos por el modelo.

c. Si bien cuando graficamos las medianas se puede apreciar un número bastante bien definido, cuando vemos las imágenes por separado, estas son bastante diferentes entre sí, como se aprecia en la Figura 3 (más abajo).

d. El hecho de que el dataset se encuentre compuesto por imágenes complica la exploración, primero porque debe buscarse un método para visualizarlo, en función del tamaño de la imágen, hay presencia de muchísimos atributos y esto puede aumentar el requerimiento de cómputo si no se restringe la cantidad de los mismos. En principio resulta difícil determinar qué píxel sería más relevante que otro. En términos de la exploración de los datos, fue necesario desde un principio escribir funciones auxiliares que permitieran la visualización de, por ejemplo, una entrada del dataset. Este paso extra comprende un obstáculo más a la hora de buscar tendencias entre los datos a simple vista.

e. Comentarios adicionales sobre la fase de Análisis Exploratorio:

Durante el análisis produjimos algunos derivados del dataset, como por ejemplo una versión normalizada, o un dataset con las medianas de cada dígito. Esto nos permitió darnos una idea de cómo estaban representados, qué pixeles eran más frecuentemente pintados y cuáles no, y dónde podríamos encontrar patrones que nos permitieran distinguir uno del otro.

Una de las primeras cosas que observamos es que había dígitos los cuales tenían un bajo valor de máximos, por ejemplo 80 cuando otros tenían 255, por lo que comprendimos que la presión del trazo afectaba los valores. Al normalizar cada fila de manera individual se eliminó la intensidad del trazo como variable ya que lo que nos interesa clasificar es la forma. Esta normalización además permitió reducir en cierta medida la niebla.

Al imprimir los valores de las medianas de cada dígito, pudimos apreciar que en general los dígitos estaban dibujados en el medio de la imagen (Figura 3). A raíz de esto, probamos hacer un recorte de todos los bordes. Sin embargo aún teníamos muchos

atributos y no resolvía nuestro principal problema que era reducir la cantidad a unos pocos, idealmente.

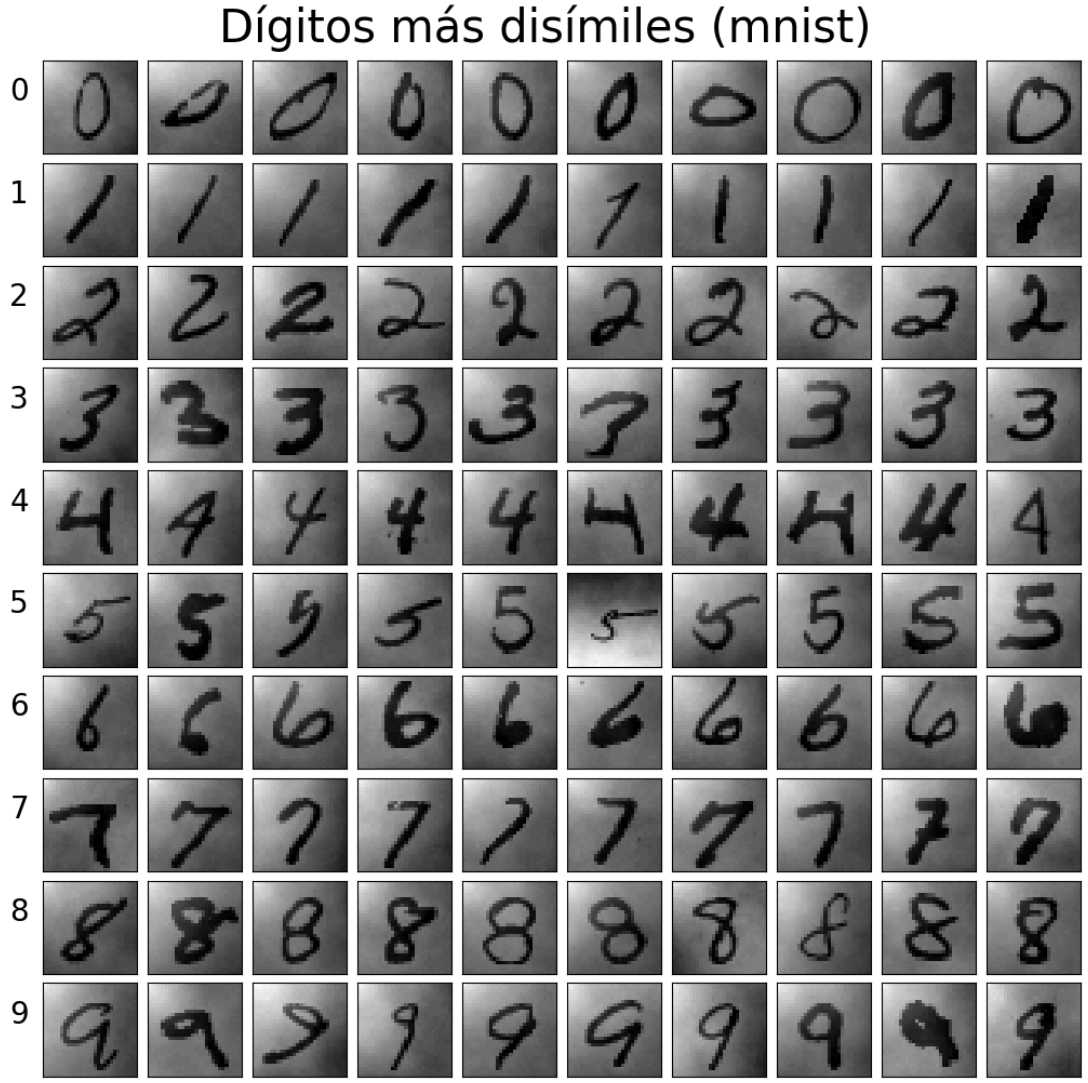

Figura 3. Las entradas «más disímiles» por cada dígito. Se buscaron las imágenes más diferentes a la mediana de su dígito correspondientes; se usó como criterio la suma del cuadrado de las diferencias entre cada píxel y su valor en la mediana del dígito correspondiente.

3. Clasificación Binaria

Luego del análisis exploratorio de la sección anterior, nos propusimos responder una de las preguntas del enunciado:

Dada una imágen perteneciente al dataset:

¿La imagen corresponde al dígito 0 o al dígito 1?

Se filtró del dataset las filas etiquetadas como “0” y “1” en un nuevo sub-dataset. Según lo observado en el análisis exploratorio, la clase con mayor frecuencia es el “1” con 7877 apariciones, y el “0” con 6903. Porcentualmente, hay un 14% más unos que ceros.

Se separaron los datos en conjuntos de train y test, tomando cuidado de estratificar para conservar la prevalencia de los unos sobre los ceros. Se comenzaron distintas pruebas de modelos sobre estos dos conjuntos. Primero se probó utilizar todos los atributos disponibles con distintos valores de k, alcanzando un máximo de 99.83% con k=2.

Tres criterios diferentes fueron desarrollados para ordenar los píxeles según su capacidad de distinguir entre los ceros y los unos (que llamaremos píxeles «discriminantes»). Como primer criterio, se partió de la diferencia entre la mediana de los unos y la de los ceros, elevada al cuadrado (para considerar todos los píxeles diferentes como positivos). Luego, a cada píxel de esta imagen derivada se lo dividió por el producto entre la varianza de ese píxel en los unos, con la varianza de ese píxel en los ceros. Esto cumple el efecto de favorecer píxeles que varían poco en ambos conjuntos pero que difieren entre las medianas. Como segundo criterio, se utilizó la varianza del conjunto en su totalidad, considerando que los píxeles que más varían son los que distinguen entre ceros y unos. Por último, el tercer criterio consistió sólamente en el segundo paso del primer criterio: la inversa del producto entre las varianzas en los ceros y las varianzas en los unos. En la Figura 4 se representan los valores de cada criterio – los valores más altos indican píxeles más discriminantes según ese criterio.

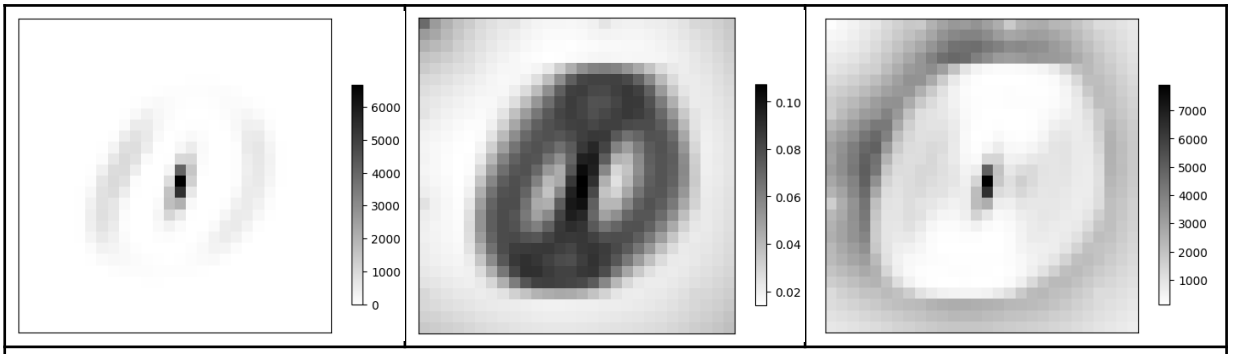

Figura 4. Imágenes producidas por los criterios 1 ‘default’ (izq.), 2 ‘varianza_total’ (cen.) y 3 ‘varianza_producto_inv’ (der.). Se observa que los píxeles del centro del uno y los del borde del cero son los más discriminantes en general.

Se evaluó qué conjuntos de 3 atributos produjo cada criterio. Se encontró que estos llegaron a un concenso en 2 de los 3 atributos, correspondientes a los píxeles centrales.

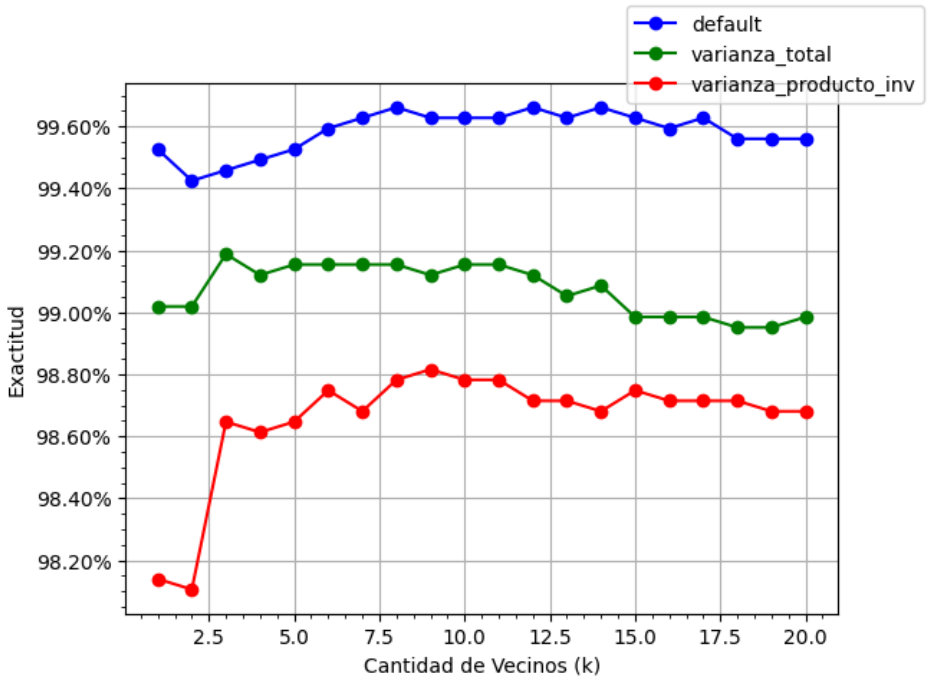

Figura 5. Desempeño de modelos KNN en función del número de vecinos K, usando sólamente los 15 píxeles más discriminantes según cada criterio. En comparación, eligiendo atributos al azar rindió una exactitud promedio máxima de 86%.

Luego de probar con diferentes valores de profundidad máxima y número de píxeles discriminantes (usando el mejor criterio, el 1), se encontraron parámetros que producen una exactitud del 99.66%: (n=15 y k=8). A comparación de usar todo el dataset, usar solo 3 atributos hace al modelo computacionalmente mucho más eficiente y escalable mientras que la exactitud que se pierde (0.17 puntos porcentuales) es mínima.

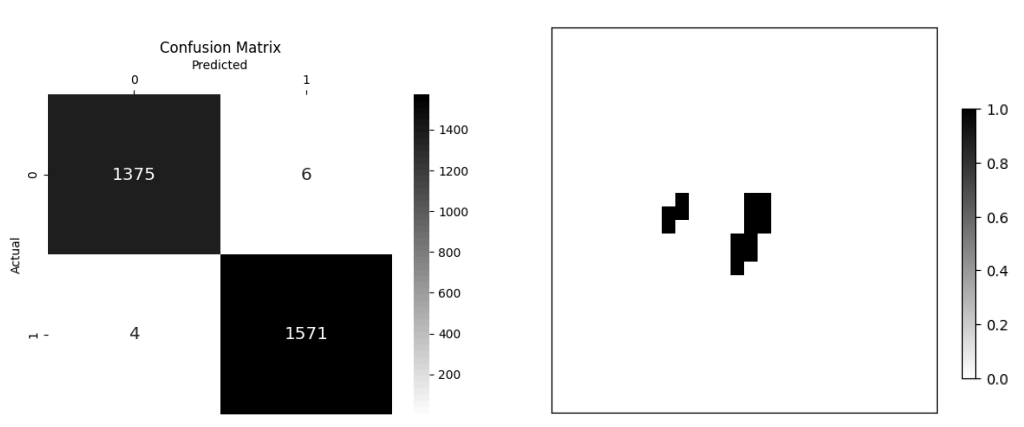

Figura 6: Izq: Matriz de confusión del modelo final, entrenado utilizando sólo los 15 atributos más discriminantes según el criterio 1. Der: Ubicación de los 15 atributos considerados como los más discriminantes por el criterio 1.

4. Clasificación Multiclase

El entrenamiento de un clasificador binario en la etapa anterior permitió reconocer (dado un 0 o 1 perteneciente al dataset) a qué clase dicha imagen pertenecía, con una alta tasa de éxito. Ahora, entonces, nos proponemos:

Dada una imágen perteneciente al dataset:

¿A cuál de los 10 dígitos corresponde la imagen?

Para iniciar, separamos el conjunto de datos en una partición de desarrollo (90%) y *held-out* (10%). Esta última será apartada hasta el final de esta sección.

Iniciamos entonces la exploración de modelos, en un inicio solo variando la profundidad, buscando maximizar la precisión:

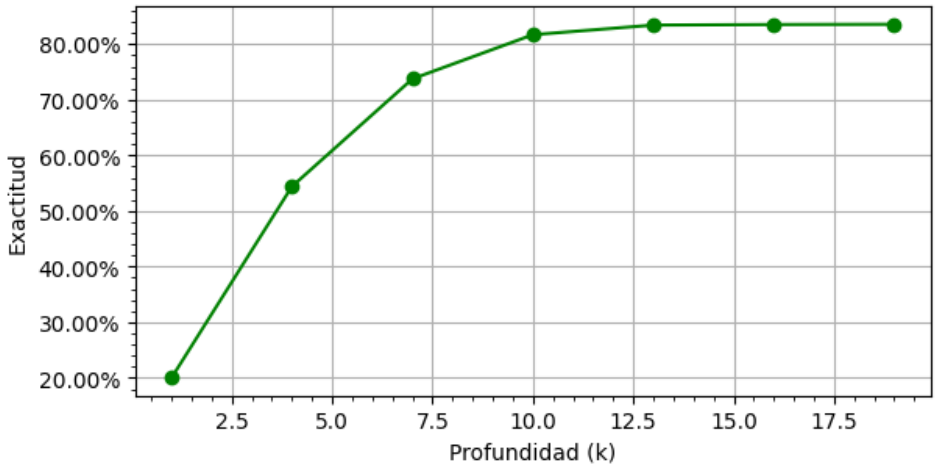

Figura 7. Precisión en función de la profundidad (k). Puede verse un claro aumento de la precisión en base al incremento de la profundidad hasta cierto punto.

Observando este incremento, fijamos la profundidad y exploramos el núm. máximo de atributos (max_features, mf). Así, evaluamos la precisión, esta vez variando la cantidad de mf, y a su vez, observando su desempeño con solo los valores del medio (cropped):

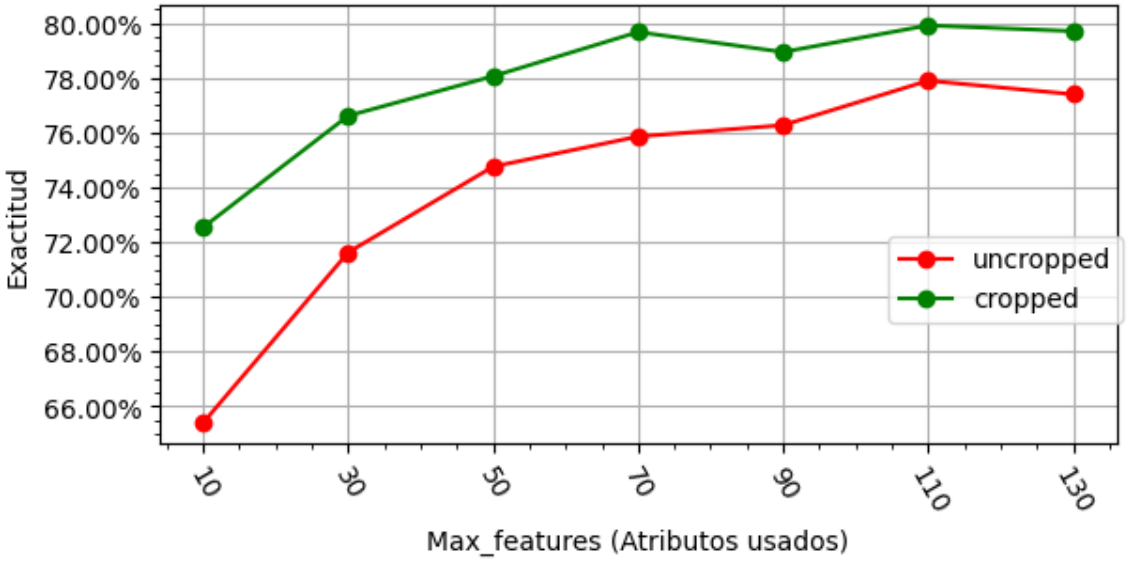

Figura 8. Exactitud en función de mf, utilizando el dataset con y sin recorte.

Vemos que al realizar una selección de los atributos del centro, aumenta el desempeño general del modelo para mfs bajos, y que mf=30 es un buen balance entre exactitud y performance (menos atributos implica menor cantidad de cálculos a realizar). Seleccionamos esta cantidad de atributos para continuar con la selección de modelos.

Evaluamos el comportamiento de la exactitud, esta vez considerando dos medidas de impureza (gini o entropía). Iniciamos por evaluar la variación de la exactitud en base de la profundidad, lo cuál resultó en un comportamiento casi idéntico para ambas. Es decir, si uno considera la totalidad de atributos y sólo varía la profundidad, ambos criterios son muy similares. Entonces procedimos a comparar su comportamiento al variar el núm. máximo de atributos:

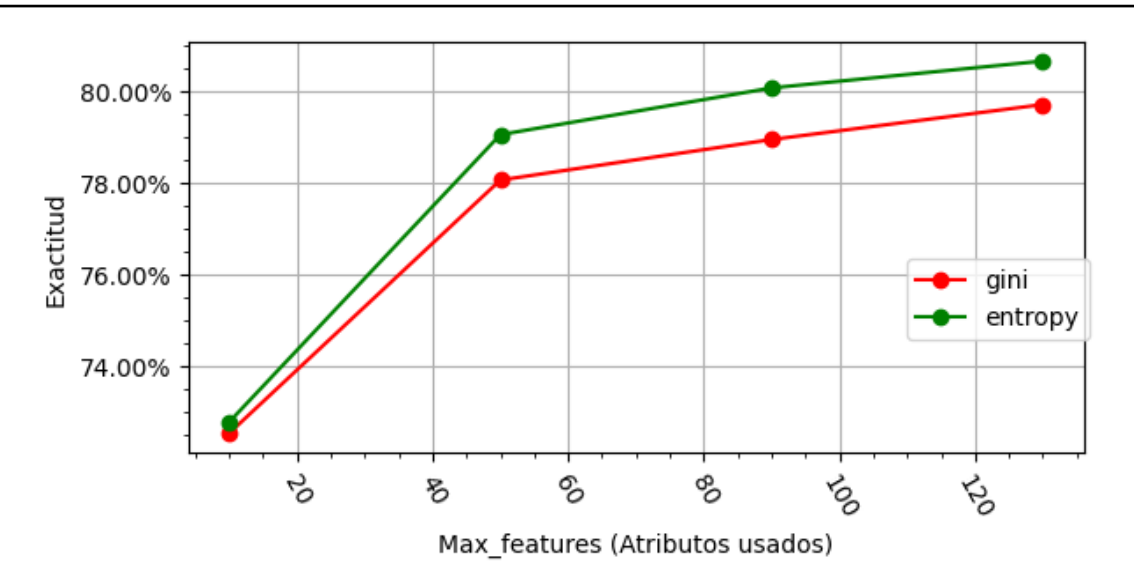

Figura 9. Exactitud en función de mf, utilizando gini o entropía como medida de impureza. Modelos entrenados a partir del dataset croppeado. Vemos que entropy rinde aproximadamente un punto más que gini, pero toma casi el doble del tiempo.

Por cuestiones de performance, y considerando que para maximizar la exactitud se entrenará el modelo final sobre la totalidad de los atributos, se eligió el criterio gini para el resto de los ensayos de la sección.

Para continuar la evaluación, y basándonos en los resultados previos, utilizamos gini, max_depth=10 y un mf=30 (con tal de acelerar el cálculo), para variar los siguientes hiper-parámetros:

- Mínimo de muestras requeridas para separar un nodo interno (min_samples_split, mss)
- Mínimo de muestras requeridas para ser una hoja (min_samples_leaf, msl)

Si bien en principio vimos una variación al probar distintos valores para ambos hiper-parámetros, encontramos que al aumentar los mf a 130, dejó de ser relevante el valor para mss y msl (el comportamiento de la exactitud casi no variaba según estos hiperparámetros). De esta forma, concluimos que el único hiper-parámetro relevante, al menos en este caso, es la profundidad. Así, evaluamos un último modelo, con los mejores valores encontrados: criterio “gini”, max_depth= 10, mss=2, msl=1 (predeterminados). Finalmente, el modelo obtuvo una métrica de exactitud de 81.70%, y la siguiente matriz de confusión:

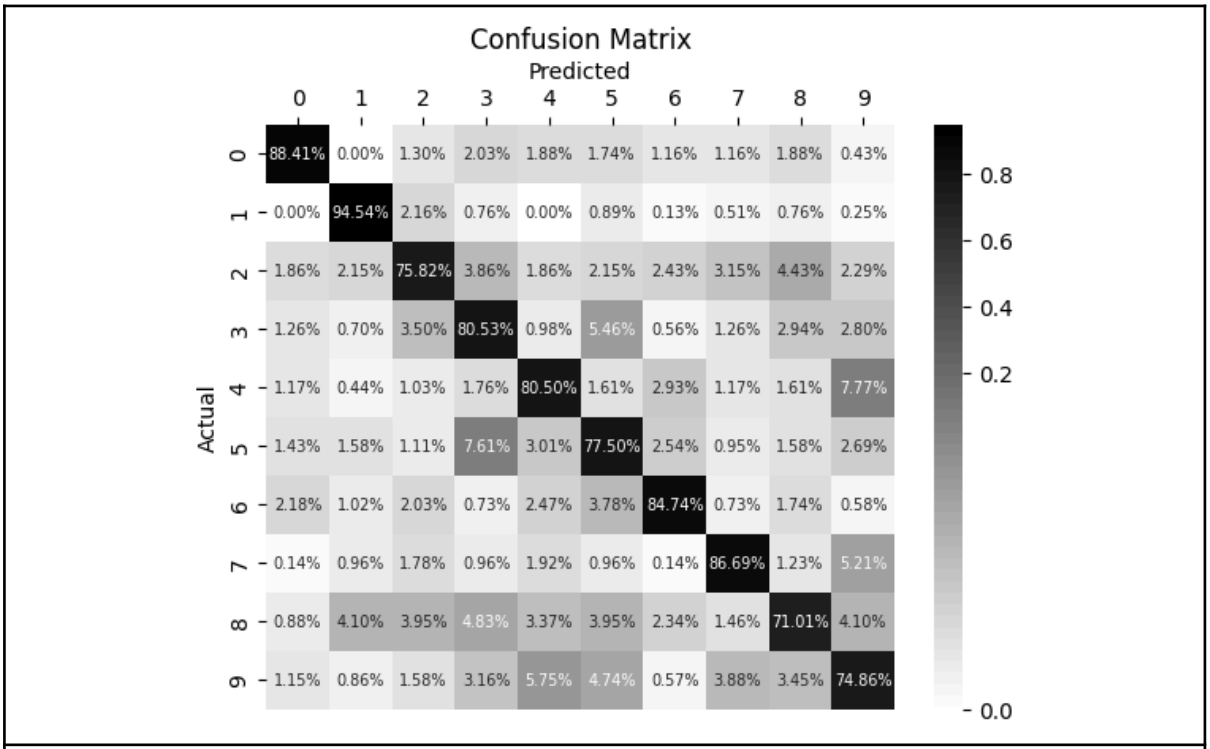

Figura 11. Matriz de confusión «Relativa» del modelo entrenado. Se dividieron los valores de la matriz de confusión original por el máximo de cada fila, con tal de que todos los valores de las filas suman 100%. Así, se puede afirmar que un 7.77% de los 4 fueron clasificados como 9. Se usó un mapa de colores de doble pendiente, para acentuar las confusiones.

5. Conclusiones

Los modelos de clasificación como el KNN y el Decision Tree Classifier son excelentes herramientas para clasificar de manera precisa, pero se deben tener en cuenta ciertas limitaciones. Cuando se trata de una clasificación binaria, con muy pocos atributos discriminantes podemos clasificar grandes volúmenes de datos. Al tener que clasificar imágenes multiclase surgen más dificultades con estos modelos, y alcanzar un nivel alto de exactitud no es tarea sencilla.

Al analizar imágenes en vez de variables categóricas, hizo más difícil clasificar y hubo que utilizar algunas estrategias, como normalizar, buscar atributos claves, como aquellos con mayor varianza en la muestra o los que fueran más discriminantes entre una clase y la otra para poder tener un valor de exactitud elevado.

En el caso de las imágenes de unos y ceros, utilizando un modelo KNN y el criterio 1, bastó con 3 píxeles para obtener aproximadamente un 98.92% de exactitud. Si utilizamos 15 atributos, podemos subir ese valor de exactitud a 99.7% aprox. sin comprometer mucho el uso de poder de cómputo, por lo que estos modelos son precisos y performantes para realizar clasificaciones binarias.

Para la clasificación multiclase comenzaron a verse las limitaciones de los árboles de decisión, incluso tomando una gran cantidad de atributos y permitir al árbol decidir los atributos. Se pudo conseguir un 81.7% de exactitud utilizando el dataset completo, manteniendo la profundidad del árbol en 10 (siendo este el máximo especificado).

El hecho de evaluar imágenes produjo que el árbol de decisión perdiera su característica de alta interpretabilidad a la hora de clasificar píxeles, ya que el número de columna no es exactamente un valor claro para el que lo lee. Probablemente un árbol de decisión no es un buen modelo para clasificar imágenes multiclase. Es más, no sería arriesgado estipular que un modelo de árbol de decisión tendría buena exactitud en el caso binario (visto que con unas pocas columnas se puede distinguir entre dígitos), y que un modelo KNN se ajustaría mejor a las particularidades del problema multiclase.