Laboratorio de Datos
2do cuatrimestre 2024

Trabajo Práctico 2

Grupo: Another Level

Integrantes: Bergaglio Pedro, Da Silva Minas Tomas y Kiss Maria Delfina

Introducción

En el presente trabajo práctico, trabajaremos con el dataset TMNIST, que es un conjunto de datos de imagen de 10 dígitos distintos (0 al 9) en diversas tipografías. Nuestro objetivo se centra en la correcta asociación de la imagen al dígito que representa sin importar la tipografía en la que se encuentre. Para ello, comenzamos realizando un análisis exploratorio de los datos, luego una clasificación binaria y por último una clasificación multiclase.

Análisis exploratorio

Para comenzar con el trabajo, realizamos un análisis detallado del dataset para entender su organización y la relevancia de algunos atributos, así como ver si existen patrones entre ellos.

Para comenzar con nuestro análisis notamos que el dataset presenta 786 atributos distintos, dos de ellos correspondientes al dígito representado y la tipografía y los 784 restantes representan la intensidad de cada píxel, representadas con un int64, que varía entre 0, blanco, y 255, negro. Como para nuestro trabajo la tipografía no es relevante, eliminamos dicho atributo. A su vez, observamos que el dataset contiene 29900 filas, es decir 29900 imágenes distintas. Entre todas ellas observamos que en el atributo que especifica el dígito de la imagen hay 10 dígitos distintos, del 0 al 9, y cada dígito posee 2990 imágenes distintas, es decir el dataset presenta una distribución simétrica de imágenes por dígito, es balanceado.

A continuación, buscamos si algunos de los atributos son más relevantes que otros, es decir si hay píxeles que son más importantes a la hora de identificar a qué dígito corresponde la imagen que otros. Para ello decidimos realizar un mapa de calor que presente para cada píxel la intensidad acumulada por todas las imágenes, podemos observarlo en la Figura 1.

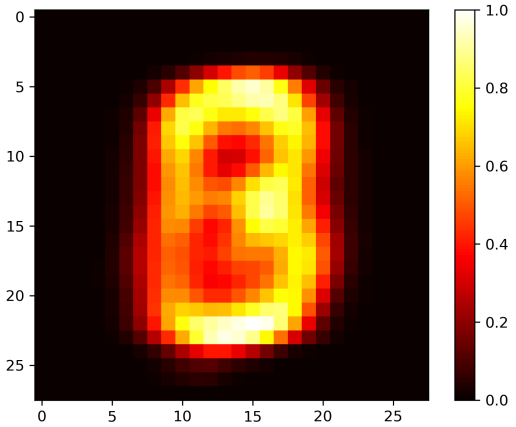

Figura 1. Mapa de calor de intensidad de pixeles.

Notamos que la mayor variación de los píxeles, que determinan el dígito de la imagen, se encuentra en el centro de la imagen. Sabemos que aquellos pixeles que tienen intensidad 0, son blancos, y están muy presentes en los bordes de la imagen, dichos no nos

dan información sobre el dígito que representa. Por lo que para proseguir con nuestro análisis eliminamos todos aquellos píxeles que la suma para todas las imágenes fuese 0, es decir que en todas las imágenes en ese píxel tienen intensidad 0.

Para continuar, decidimos buscar si existen números similares entre sí. Comenzamos graficando en un scatter plot las diferencias absolutas para cada de los pixeles de dos a dos dígitos.

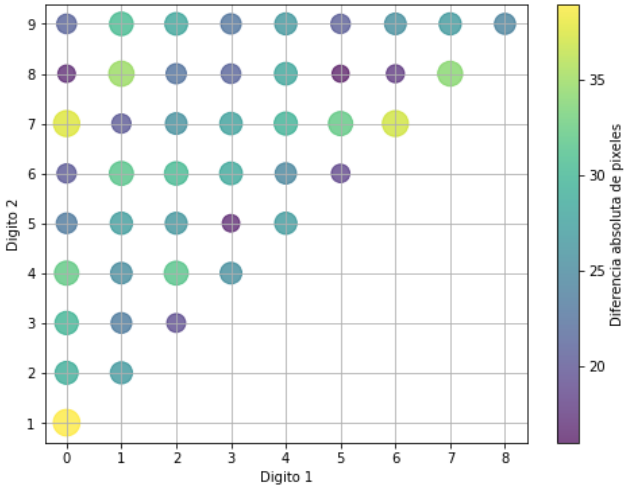

Figura 2. Scatter plot de las diferencias absolutas entre píxeles entre dígitos.

Podemos observar en la Figura 2 que los dígitos que más difieren entre sí son el 0 con el 1 y los más similares entre sí el 5 y el 8. Para continuar con el análisis decidimos seleccionar los números 1, 3 y 8. Comenzamos observando una imagen correspondiente a cada uno de los tres dígitos, representados en la Figura 2.

Figura 3. Imágenes de tipografías aleatorias para los dígitos 1, 3 y 8.

Como podemos notar en la Figura 3, hay números que son más fáciles de diferenciar entre sí que otros. Por ejemplo, entre el número 8 y 3 hay bastante similitud en la ubicación de los píxeles que definen la silueta del dígito, en cambio, entre el dígito 1 y el 3 es una diferencia bastante más marcada en cuanto a los píxeles que representan el dígito, hay una menor cantidad de píxeles activados en su contorno. Para ver más claramente esta diferencia realizamos un mapa de calor, superponiendo los píxeles promedio para cada dígito, del 1 con el 3 (Figura 4 A) y del 3 con el 8 (Figura 4 B).

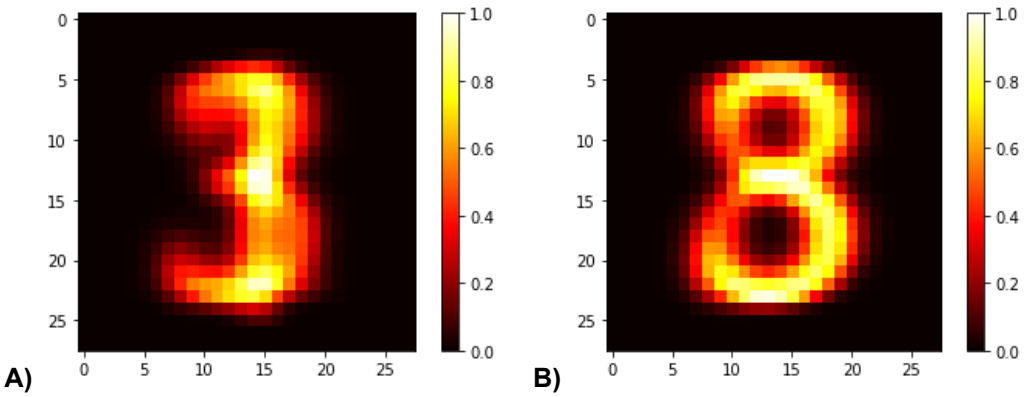

Figura 4. Mapa de calor para comparar dos dígitos. **A)** Dígito 1 y 3. **B)** Dígito 3 y 8.

Podemos notar que en la Figura 4 B) comparada con la Figura 4 A) los píxeles son muy similares y no se distinguen entre el dígito 3 y 8, en cambio en la Figura 4 A) es más notoria la diferencia entre ambos dígitos. A su vez, decidimos graficar el valor absoluto de la diferencia entre ambos píxeles.

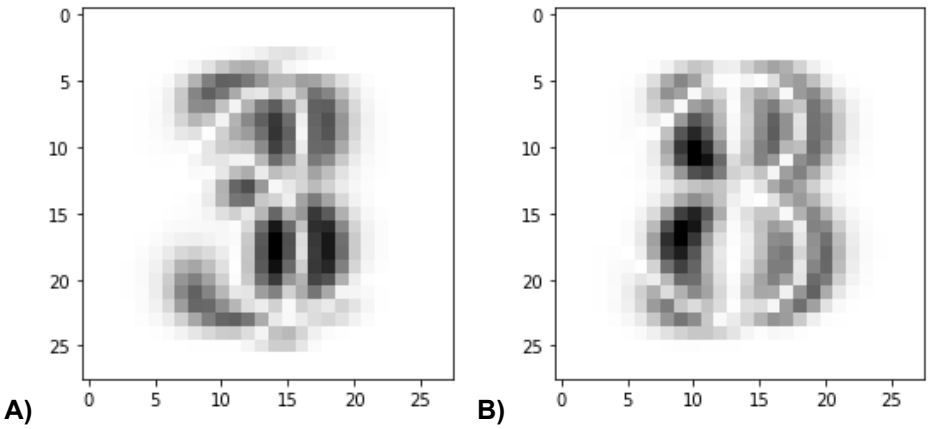

Figura 5. Diferencia absoluta de los píxeles entre dos dígitos. **A)** Dígito 1 y 3. **B)** Dígito 3 y 8.

En la Figura 5 A) y B) podemos observar la diferencia absoluta entre las intensidades de dos dígitos para cada pixel. A mayor oscuridad del pixel, más difieren esos pixeles entre los dos dígitos. Observamos que entre los dígitos 3 y 8, Figura 5 B), hay menores disparidades de píxeles a comparación de los dígitos 1 y 3 Figura 5, A).

Para proseguir, nos preguntamos si para un mismo dígito todas las imágenes son muy similares entre sí, para ello seleccionamos el dígito 0 y observamos cómo cambiaba de acuerdo a la tipografía. Como podemos observar en la Figura 6, si bien en todas las imágenes podemos distinguir el dígito, de una imagen a otra la silueta de dicho varía en gran medida. Realizamos un nuevo mapa de calor, en este caso para el dígito 0, Figura 7, en ella podemos distinguir claramente la silueta del dígito. Por lo que si bien todas las imágenes pueden no ser muy similares, los pixeles que ayudan a determinar a qué dígito corresponde si son parecidos entre sí.

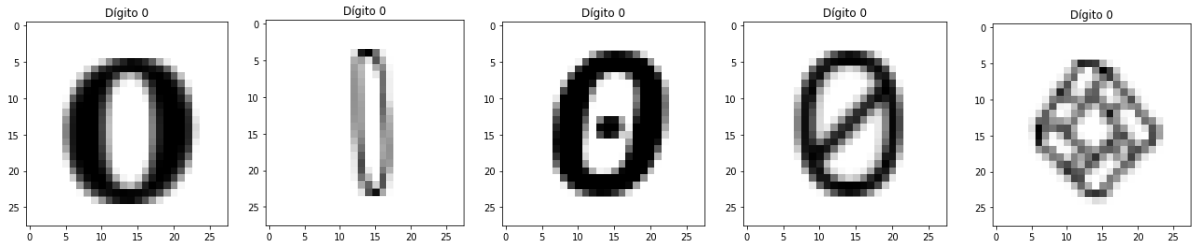

Figura 6. Dígito 0 en diversas tipografías.

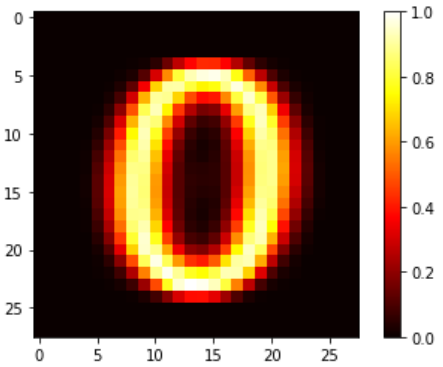

Figura 7. Mapa de calor del dígito 0.

En general, este análisis exploratorio reveló que trabajar con un dataset de imágenes requiere técnicas y enfoques distintos en comparación a los datasets con los que veníamos trabajando normalmente, como el que utilizamos en el Trabajo Práctico 1. Por lo que tuvimos que modificar un poco nuestra forma de trabajo ya que las imágenes requieren otras técnicas de trabajo para entender cómo se caracteriza el dataset. Esto nos resultó un poco complicado de entender al inicio pero no se nos hizo más difícil en general. Al mismo tiempo, por ejemplo para el Trabajo Práctico 1, hemos trabajado con datasets con grandes complejidades e incluso muy mala calidad, en este caso los datos están en perfectas condiciones de limpieza y no requieren pasos previos relacionados a esto, además el dataset es extenso pero simple: una columna ‘name’ con el tipo de letra, ‘label’ con el número correcto, y luego los pixeles.

Clasificación binaria

Para comenzar a adentrarnos más en el problema del trabajo, comenzamos tomando un problema simplificado, la clasificación binaria entre el dígito 0 y el dígito 1. Queremos a partir de una imagen que represente alguno de los dos dígitos poder determinar claramente a cual representa. Para ello usamos el algoritmo KNN, k nearest neighbors, que lo que hace es considerar los k valores más cercanos al nuevo valor y promedia los k valores más cercanos al nuevo valor. Vamos a ir ajustando diferentes atributos y números de vecinos, y para cada uno iremos evaluando el rendimiento de los modelos en un conjunto de datos de prueba.

Para comenzar, filtramos nuestro dataset original quedándonos únicamente con aquellas imágenes que tengan como label al dígito 0 o al dígito 1. Estudiando este nuevo data frame, notamos que tanto para el dígito 0 como para el dígito 1 tenemos 2990 imágenes distintas, por lo tanto está balanceado respecto a estas dos clases. Luego,

separamos el conjunto de train y de test desde el nuevo data frame. Decidimos utilizar el 80% para train y el 20% para test.

Paso siguiente, comenzamos a entrenar el modelo de KNN usando en principio 3 atributos. Para ello, elegimos los conjuntos de 3 atributos de tres formas diferentes: a mano, los 3 pixeles con mayor diferencia absoluta y al azar a partir de aquellos pixeles que tenían intensidad distinta de cero para alguna de todas las imágenes del dataset.

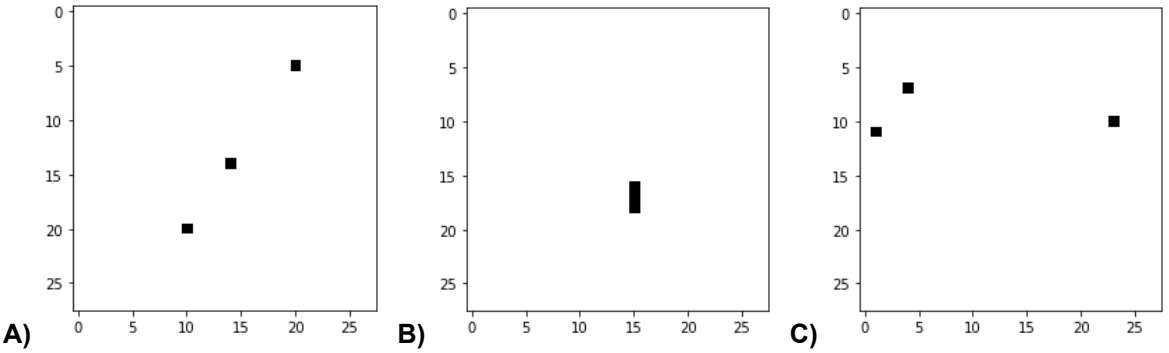

Figura 8. Conjunto de tres atributos seleccionados. **A)** a mano, **B)** mayor diferencia y **C)** aleatorio.

Para cada una de las selecciones realizadas ilustradas en la Figura 8, para el caso C) graficamos los tres píxeles que mejor exactitud nos dieron. Obtuvimos que la exactitud de la elección a mano era mejor que de acuerdo a la diferencia absoluta y en algunos casos también fue mejor que la exactitud al seleccionarlos de manera aleatoria.

Para continuar decidimos probar qué sucede si variamos la cantidad de atributos seleccionados, para ello elegimos 50 conjuntos de 3, 5, 7, 9, 11 y 13 atributos respectivamente de manera aleatoria, entrenamos el modelo y para cada conjunto de diferente cantidad de atributos calculamos el promedio de exactitud.

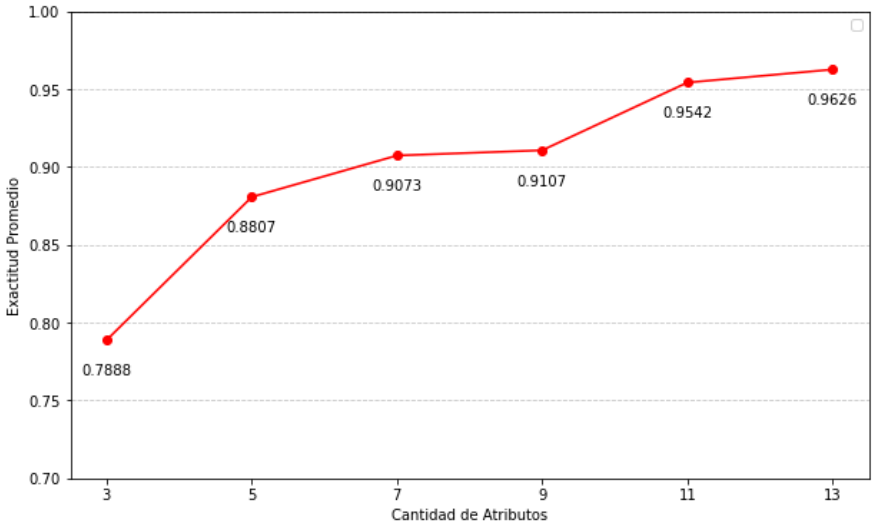

Figura 9. Variación de la exactitud respecto a la cantidad de atributos para 3 vecinos.

Podemos observar en la Figura 9 que a mayor cantidad de atributos mayor es la exactitud promedio del modelo. Por último, para ver cómo afecta al modelo variar la cantidad de vecinos, además de variar la cantidad de atributos, variamos también los k y

observamos cómo varía la exactitud. Nuevamente entrenamos el modelo para cada combinación de hiperparametros.

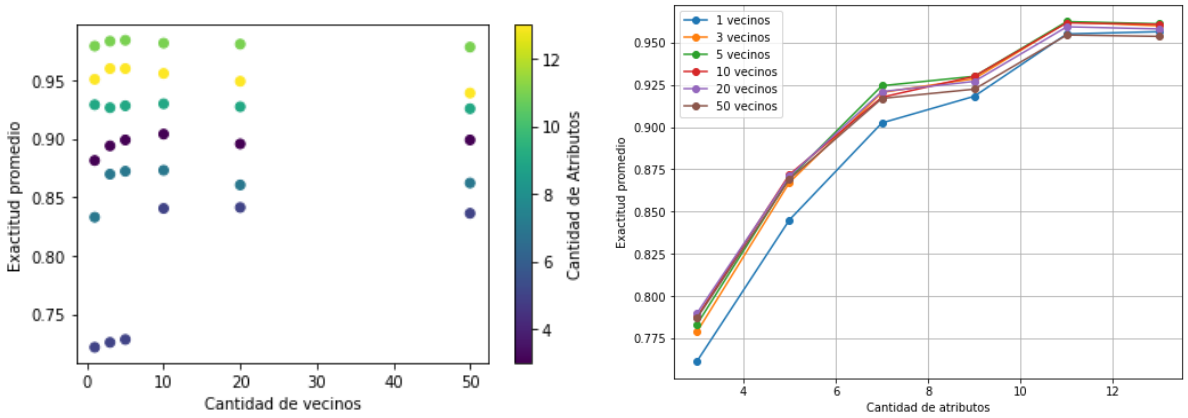

Figura 10. Comparación de la exactitud promedio, cantidad de vecinos y atributos usados.

A partir de la figura 10 observamos que el modelo con mejor exactitud que obtuvimos fue aquel que entrenamos con 11 atributos y 5 vecinos. Sin embargo, al analizar individualmente cada combinación sin promediar los resultados, luego de probar distintas veces con diversos datos aleatorios, identificamos que la combinación óptima depende de los datos aleatorios elegidos, pero se repitió la combinación de 13 atributos con 5 vecinos, lo que sugiere que, en algunos casos, una mayor cantidad de atributos puede mejorar la robustez del modelo frente a variaciones en los datos. A pesar de esto,no logramos identificar una configuración única que siempre maximice la exactitud, al ejecutar distintas veces el modelo, donde se seleccionan los atributos de manera aleatoria, el modelo que mejor exactitud presenta fue variando siempre. En algunos casos, configuraciones con menos atributos y más vecinos resultaron ser las mejores, mientras que en otros, un mayor número de atributos con menos vecinos ofrecía un mejor rendimiento. Esto indica que la exactitud del modelo KNN depende no solo de la cantidad de atributos y vecinos, sino también de la composición específica de los datos utilizados en cada entrenamiento, lo que resalta la importancia de una cuidadosa selección de hiper parámetros para maximizar la precisión.

Clasificación multiclase

A partir de ahora, queremos tener en cuenta todos los dígitos posibles y, dada una imagen, poder identificar qué número es. Por lo tanto, nos planteamos hacer una clasificación multiclase, con árboles de decisión. Para eso separamos el conjunto de datos en dos grupos, datos de desarrollo, que representan el 80% del conjunto de datos, y utilizamos para entrenar y ajustar el modelo, y datos de validación, que representan el 20% restante y se utilizan para evaluar el rendimiento final del modelo.

Para poder encontrar el modelo más preciso, probamos distintas combinaciones de hiperparámetros. Variamos la profundidad del árbol entre 1 y 10 y también los criterios de selección entre Gini y Entropy. Para evaluar las distintas combinaciones de hiperparametros utilizando k-folding, aplicando 5 folds. Y a partir de ello obtuvimos una estimación de la exactitud promedio para cada modelo.

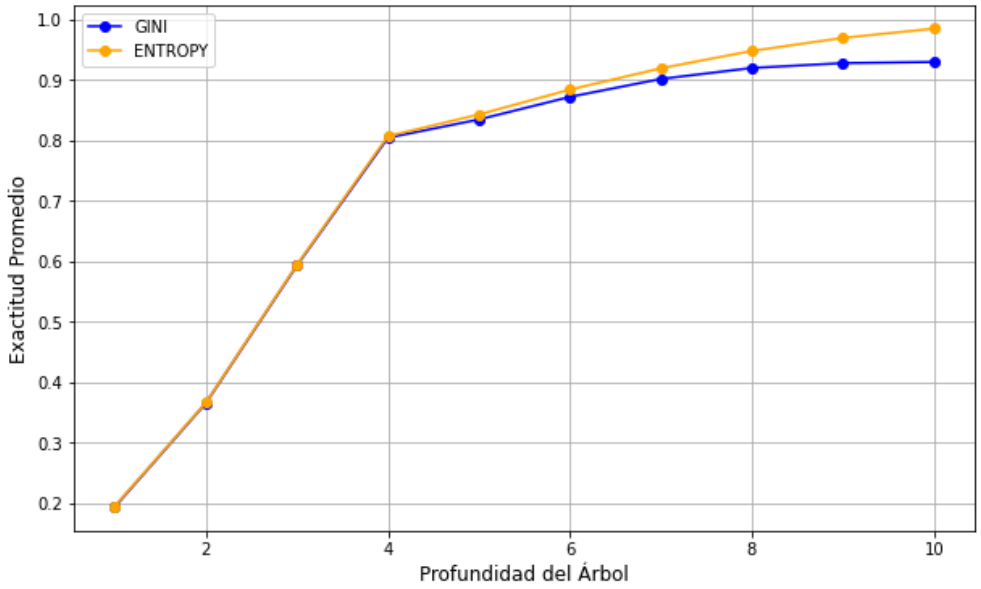

Figura 11. Comparación entre la exactitud promedio y la profundidad del árbol, para los diferentes criterios de selección.

En el gráfico mostramos el rendimiento de las distintas combinaciones de hiperparametros. Concluimos que la mejor combinación sucede al utilizar el criterio de selección entropy y la una profundidad máxima del árbol 10, obteniendo una exactitud promedio de 0,93.

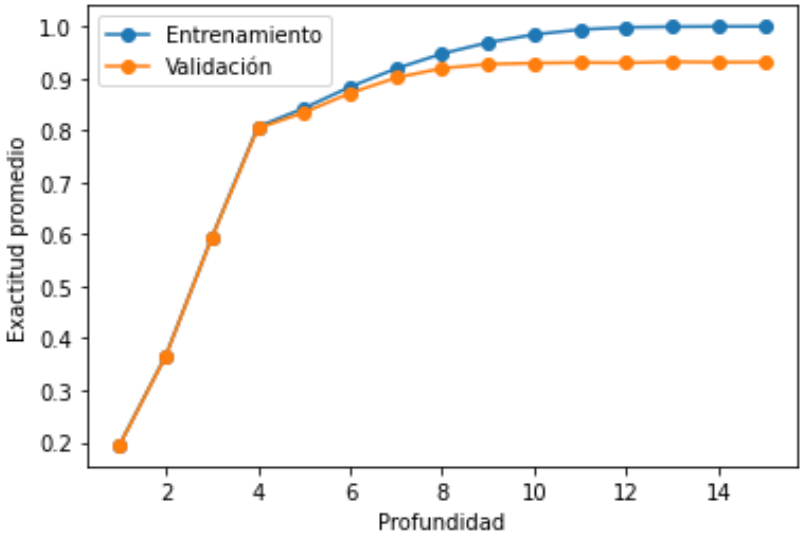

Figura 12. Comparación entre la exactitud promedio y la profundidad, entre datos de desarrollo y validación.

Una vez que tenemos la mejor combinación de hiperparametros, comparamos la exactitud entre los datos de desarrollo y los de validación. Podemos observar que el modelo se adapta muy bien a los datos de validación, pero luego de la altura 11 se puede ver que ocurre un sobreajuste, ya que el modelo empieza a memorizar los datos de entrenamiento, disminuyendo así su capacidad de generalización. Para evaluar más detalladamente el modelo analizamos la matriz de confusión.

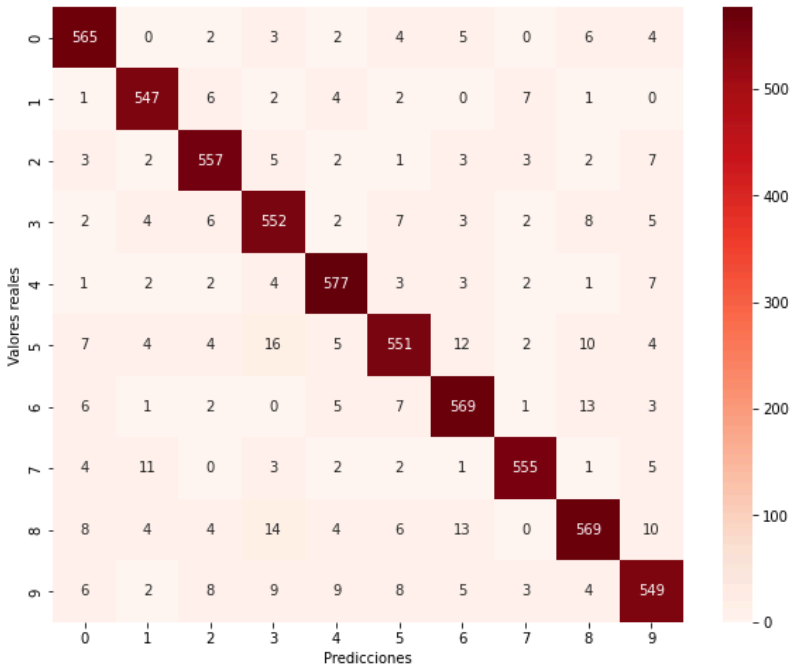

Figura 13. Matriz de confusión del modelo seleccionado.

Viendo la matriz de confusión del mejor modelo presentada en la Figura 13, podemos confirmar que el modelo posee una muy buena exactitud. Los elementos de la diagonal son las predicciones bien hechas, mientras que las demás son las incorrectas. Y podemos notar que la mayoría de las predicciones se encuentran en la diagonal principal, además, se puede ver que la matriz está bien balanceada, es decir, que el modelo está bien implementado.

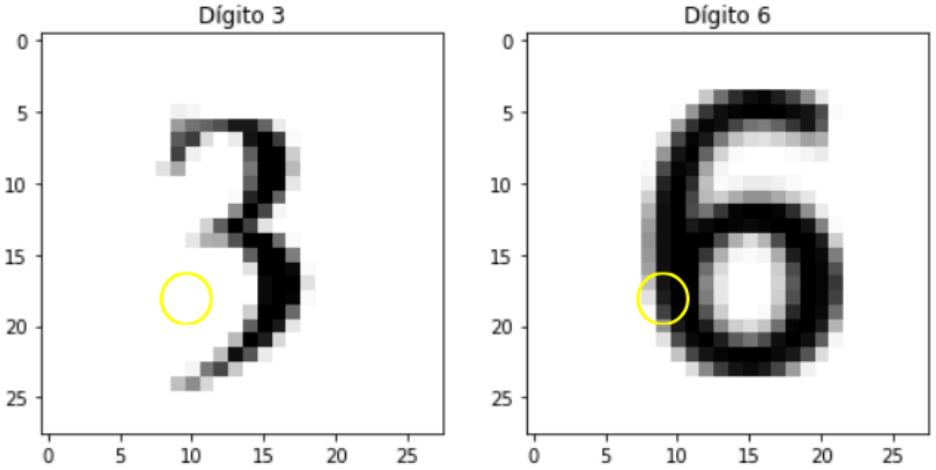

Figura 14. Diferencias entre 3 y 6.

Notamos que los dígitos en los que más errores se generan son el 5 con el 3 y el 8 con el 3. Esto puede ser ocasionado por la similitud que poseen entre sí y la distribución de los píxeles. Mientras que los dígitos que menos se confunden son el 0 con el 1, 7 con el 2 y 6 con el 3, entre otros. Esto se debe a que entre ciertos números hay diferencias muy marcadas, por ejemplo, en la figura 14 B), se puede notar que cuando el modelo se fije en el píxel que aparece coloreado en amarillo se descartara a el 6 como posible digito de esa

imagen. Las diferencias y similitudes entre los dígitos observadas en la matriz de confusión coinciden con lo que habíamos observado en la sección de ‘Análisis exploratorio’ donde habíamos notado que los dígitos que más difieren eran el 0 con el 1 y los más similares el 5 con el 3.

Conclusión

Durante el trabajo hemos utilizado diversas técnicas de análisis y clasificación al dataset TMNIST con el objetivo de poder clasificar según el dígito de las imágenes de dicho. Iniciamos nuestro trabajo explorando los datos para comprender cómo estaba organizado el dataset y entender si algunos atributos tienen más relevancia que otros. Notamos allí que los píxeles que se ubican en la frontera de la imagen no son de mucha utilidad comparados con los píxeles más bien cercanos al centro que presentan una mayor variabilidad de intensidad y ayudan a determinar de qué dígito se trata. Esta información fue posteriormente utilizada a la hora de entrenar nuestros modelos, descartando así los atributos que no sumaban información y pudiendo enfocarnos en los que sí eran de importancia.

Para continuar nos adentramos en el problema de la clasificación binaria enfocándonos en entrenar el modelo de K-Nearest Neighbors que mejor pudiese distinguir entre los dígitos 0 y 1. Para ello indagamos en diversas formas de selección de atributos y cantidad de vecinos. Observamos que el modelo presenta significativas variaciones de acuerdo a la forma en la que son seleccionados los atributos, la cantidad de atributos y la cantidad de vecinos. Nuestro análisis mostró que, aunque configuraciones con menos atributos pueden ser eficaces, aumentar tanto la cantidad de atributos como el número de vecinos mejora en general la exactitud del modelo. A pesar de esto notamos que no existe una única configuración que maximice la exactitud del modelo, pues los resultados varían de acuerdo a los datos de entrenamiento que son seleccionados de manera aleatoria.

Por último extendimos el análisis al problema de clasificación multiclase, donde buscamos poder identificar cualquier dígito entre el 0 y el 9. Para esto evaluamos diversos modelos de árboles de decisión utilizando diversos parámetros, mediante validación cruzada con k-folding. Notamos que el mejor modelo lo obtuvimos a partir del criterio de selección entropy y una profundidad máxima de 10. Además, observamos que al aumentar aún más la profundidad del árbol el modelo alcanzó un mejor rendimiento en una profundidad de 14, pero comenzó un sobreajuste en profundidades mayores reduciendo la exactitud, para continuar con el trabajo utilizamos una profundidad máxima de 10. El análisis de la matriz de confusión, generada a partir del entrenamiento del modelo con los datos de desarrollo y validación, nos permitió terminar de confirmar nuestra hipótesis previa de que ciertos dígitos, como el 3 y el 8, son más difíciles de diferenciar debido a similitudes en la distribución de los píxeles, mientras que otros, como el 7 y el 2, presentan características más distintivas, lo que facilita su clasificación.

Este trabajo práctico nos permitió desarrollar un entendimiento más profundo de cómo seleccionar atributos y ajustar hiperparámetros para optimizar el rendimiento de distintos modelos de clasificación. Pudimos entrenar nuestra clasificación y selección de modelos por medio de la validación cruzada.