Laboratorio de Datos: Clasificación y Validación Cruzada

Grupo: *Sin Nombre*		
Apellido	Nombre	LU
Wencelblat	Agustin	878/23

1 - Introduccion:

¿Cómo podemos identificar distintos dígitos escritos usando varias fuentes? ¿Existe algún algoritmo conocido que pueda hacer esto? La respuesta es que no, no existe ninguno como tal, y ese el el problema principal con el que nos encontramos en el área de aprendizaje automático. Así, utilizaremos modelos de aprendizaje supervisado para poder identificar distintos dígitos.

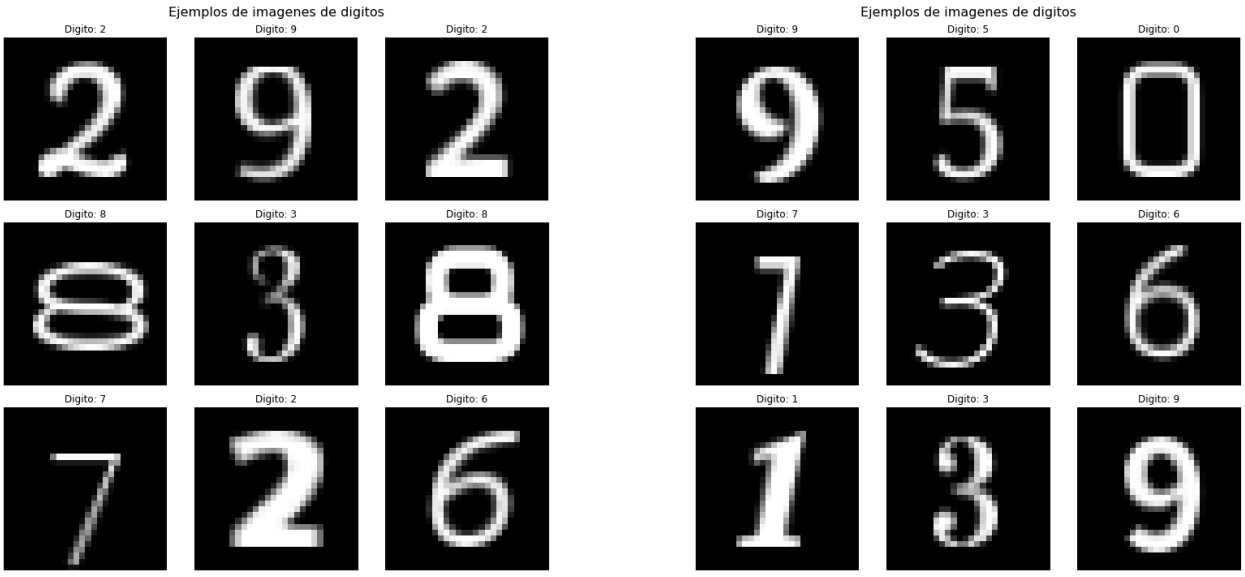
Para realizar nuestro análisis, contamos con la informacion de una base de datos, llamada TMNIST, que contiene 29900 ejemplos con labels y nombres de fuentes, y valores representando los 28x28 píxeles de la imagen correspondiente a cada dígito. Primero realizamos un análisis exploratorio de los datos, viendo que forma toman y cómo varian entre si. Luego, comenzamos con una clasificación binaria, para distinguir exclusivamente entre los dígitos 0 y 1. Finalmente, incluimos todo el dataset para realizar una clasificación multiclase, distinguiendo entre los 10 digitos posibles.

2 - Experimentos:

2.1 - Analisis Exploratorio:

La base de datos TMNIST consiste en 29.900 ejemplos con etiquetas y nombres de fuentes usadas. Cada fila contiene 786 elementos: el primero es el nombre de la fuente, el segundo un label identificando al dígito y los otros 784 se corresponden a los valores en escala de grises de la imagen de 28x28 píxeles. Cada etiqueta aparece 2990 veces, por lo que podemos decir que nuestro dataset está balanceado y contamos con 2990 fuentes distintas.

Luego, buscamos visualizar qué forma toman nuestros datos, y como podemos ver como interactua su representación con como se ve la imagen de 28x28 pixeles. Asi, elegimos algunos dígitos del dataset al azar para visualizar:



Figuras 1 y 2: Visualización de algunos dígitos del dataset TMNIST

Ahora buscamos observar qué atributos de nuestros datos son más importantes para ver a qué número pertenece la imagen. Es decir, queremos saber cuáles de los 784 valores correspondientes a cada pixel son más relevantes a la hora de identificar los distintos dígitos. Para poder visualizar usando un heatmap eso calculamos la varianza, viendo cuanto varia cada píxel entre diferentes ejemplos:

- Si la varianza es alta, eso significa que el valor de ese pixel varía mucho entre las distintas imagenes (Por ende su informacion es más importante)
- Si la varianza es baja, eso implica que su valor es más similar al de la mayoría de las imagenes (Por ende contiene información menos importante)

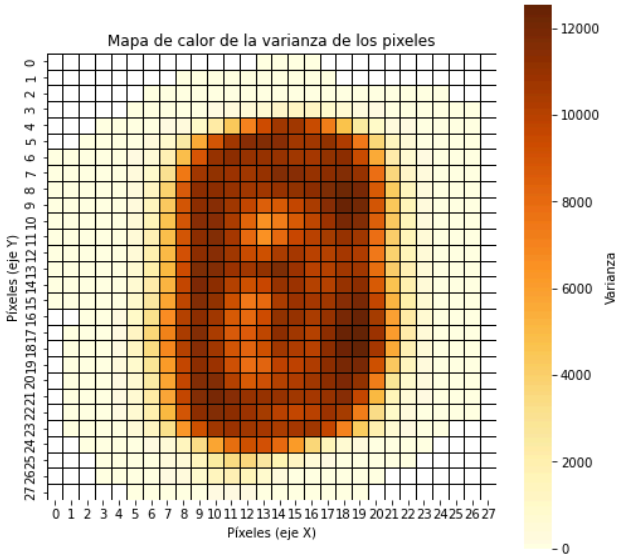


Figura 3: Mapa de calor de la varianza de los pixeles

Así, podemos observar que los valores más importantes a la hora de identificar cada uno de los digitos son los más cercanos al centro de la imagen, mientras que a medida que un píxel individual se aleja de este centro, su varianza se reduce. Podemos notar que algunos píxeles tienen varianza 0, es decir, que siempre son iguales (pues siempre son 0) lo que implica que los valores de estos pixeles no serían de valor a la hora de identificar los dígitos. Luego, armamos un heatmap con una máscara de ceros para ver qué pixeles siempre son 0.

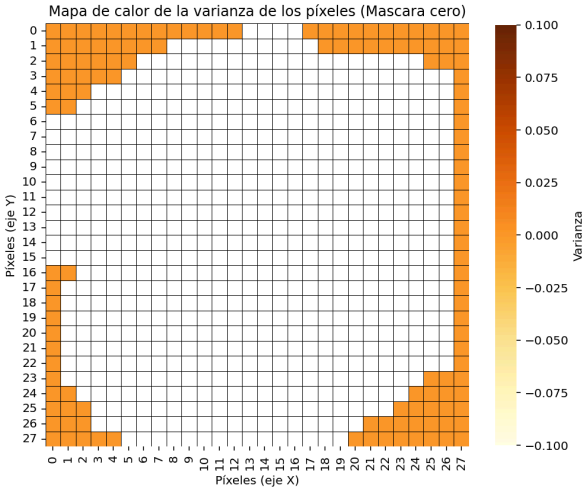


Figura 4: Mapa de calor con los pixeles que siempre son 0

Aunque podemos ver que estos datos no tendrán ningún tipo de incidencia a la hora de identificar los dígitos, no las excluimos de nuestro análisis debido a las complicaciones que esto generaría a la hora de reformar y alterar la base de datos original.

Continuando nuestro análisis exploratorio de la base de datos, queremos identificar posibles similitudes entre distintos digitos, que podrían generar problemas a la hora de identificarlos si es que son muy similares. Así, vamos calculando el “dígito promedio” de 3 grupos de dígitos para determinar visualmente si son similares. Por otro lado, también calculamos la diferencia de valor absoluto de los atributos de esos dígitos, para así poder ver donde “ocurren’ las similitudes y las diferencias entre ellos.

Figura 5: Visualización de dígitos promedio 1, 3 y 8

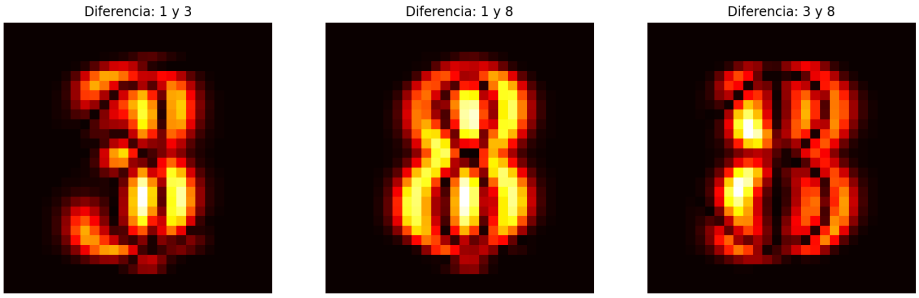


Figura 6: Diferencia de valores absolutos de dígitos 1, 3 y 8

Observando las figuras 5 y 6 podemos notar que el 1 no posee casi ninguna similitud con los dígitos 3 y 8, lo cual podemos ver claramente en la figura 6, donde los colores más intensos representan una alta diferencia absoluta entre los valores de los atributos de los dos dígitos. Por otro lado, 3 y el 8 resultan ser mucho mas similares entre si, pues comparten muchos mas valores y asi, en la figura 6, los digitos presentan una diferencia absoluta mucho menor, visualizada mediante una menor intensidad de color. Luego, continuamos nuestro análisis con los siguientes grupos.

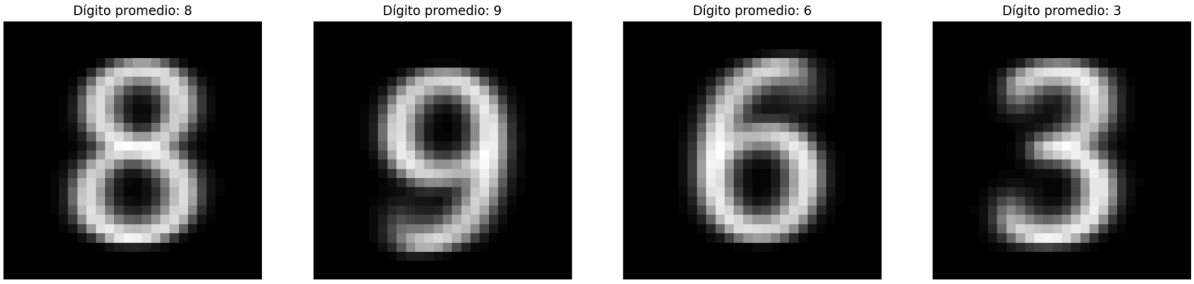


Figura 7: Visualización de dígitos promedio 8, 9, 6 y 3

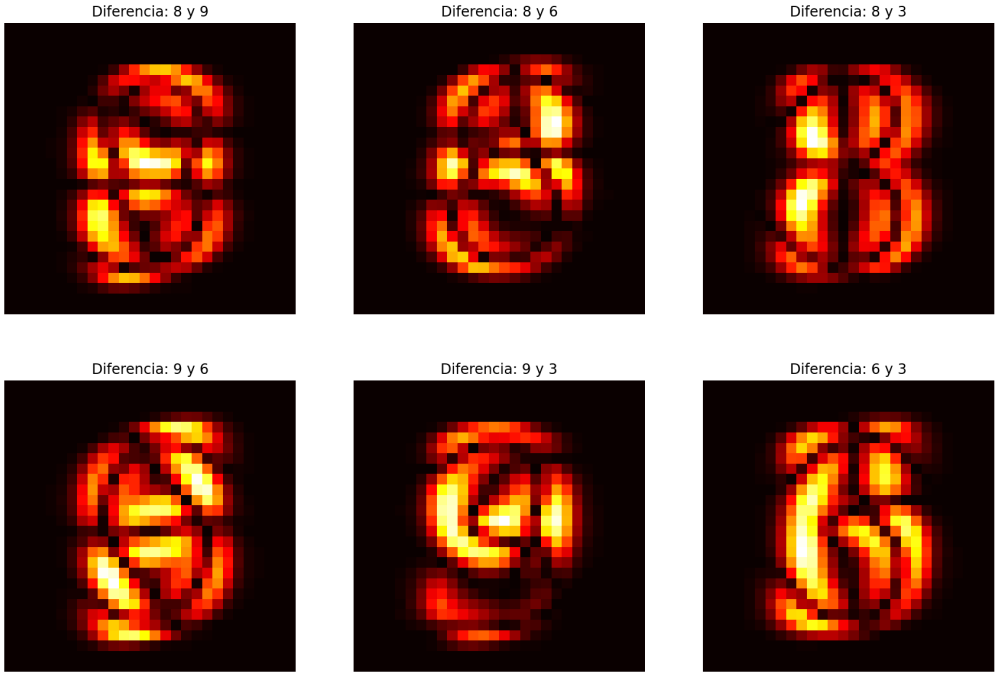
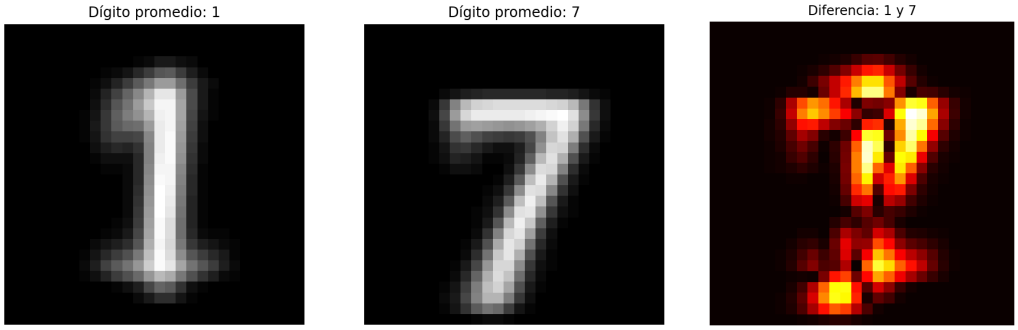


Figura 8: Diferencia de valores absolutos de dígitos 3, 6, 8 y 9

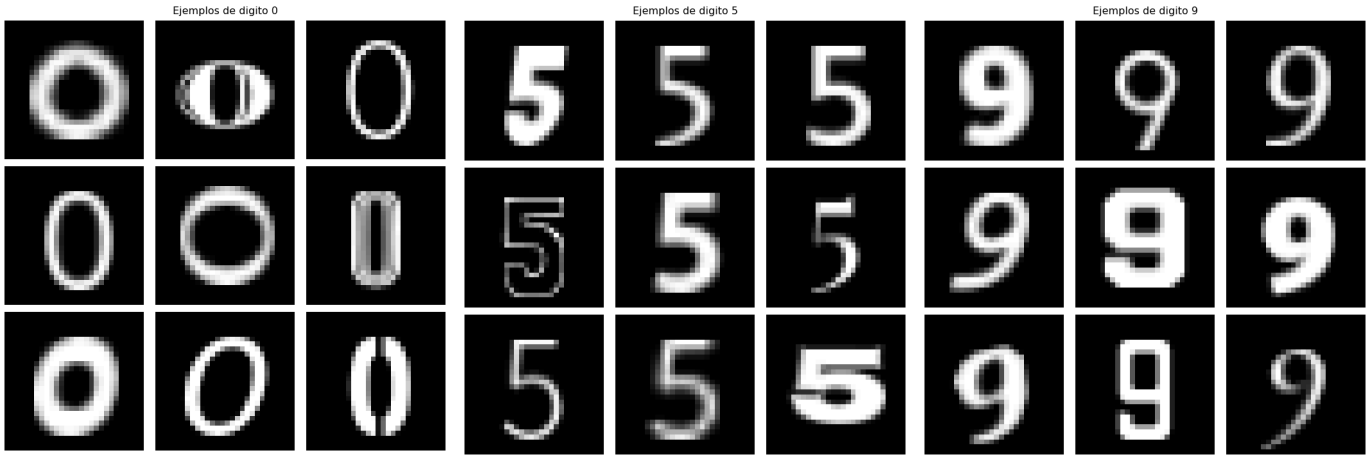
Podemos observar que los 4 dígitos tienen un cierto rango de similitudes, con algunas siendo más marcadas que otras. En la figura 8 podemos notar que los digitos con menor intensidad de color, es decir, menor diferencia de valores absolutos son el 8 y el 6, el 8 y el 3 (como vimos anteriormente) y el 9 y 3. El resto poseen algunas similitudes, pero tienen diferencias muy marcadas, especialmente el 9 y 6, pues son la misma “imagen” espejada horizontalmente y el 6 y 3, que poseen similitudes mayormente en la parte inferior derecha de sus respectivas imágenes.



Figuras 9 y 10: Visualización de dígitos promedio 1 y 7 y su diferencia absoluta

Contrario a lo esperado, los dígitos promedio del 1 y el 7 no poseen muchas similitudes, como podemos observar en la figura 10, pues poseen muchos atributos donde sus valores absolutos son sumamente distintos.

Finalmente, para concluir nuestro análisis exploratorio de los datos, vamos a observar distintos niveles de variabilidad entre un mismo dígito, es decir, queremos ver si distintas fuentes representan un dígito de formas muy distintas. Asimismo, repetimos la visualización realizada para nuestro análisis inicial de datos, solo que filtrando nuestro dataset para solo contener los numeros que queremos analizar.



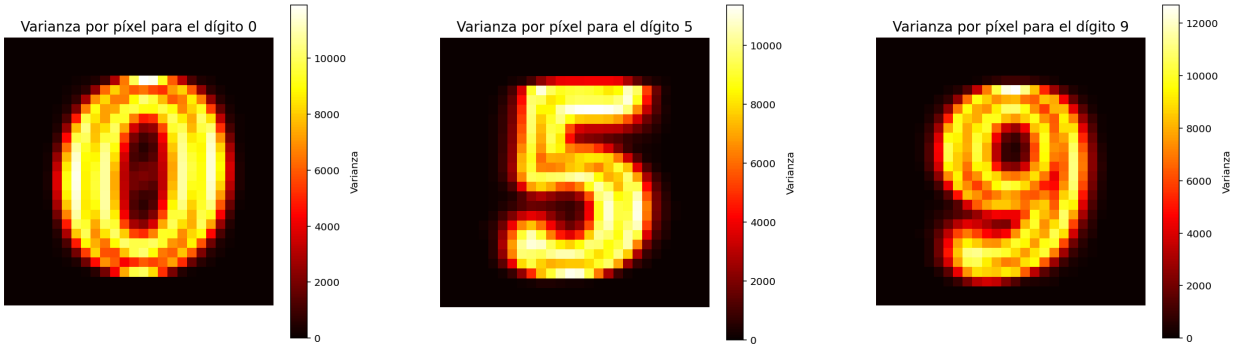
Figuras 11, 12 y 13: Visualizacion de algunos digitos 0, 5 y 9

A continuacion, marcamos algunas diferencias entre digitos que consideramos importantes:

- Existen algunas fuentes que solo incluyen el outline de un numero y no su relleno, como podemos observar en la cuarta visualizacion de la figura 12.
- Otras fuentes parecen estar en formato “Bold” como es el caso con la novena visualizacion de la figura 12 y la 5ta visualizacion de la figura 13.
- Tambien hay fuentes en italicas, que distorsionan significativamente la forma del digito, como es el caso en la novena visualizacion de la figura 13.

- En la figura 11, podemos notar que el 0 está sumamente distorsionado entre distintas fuentes, variando significativamente en altura, grosor y ancho.

Por otro lado, para entender mejor cómo varían los dígitos en sí mismos, visualizamos sus varianzas, al igual que hicimos anteriormente para todo el dataset.



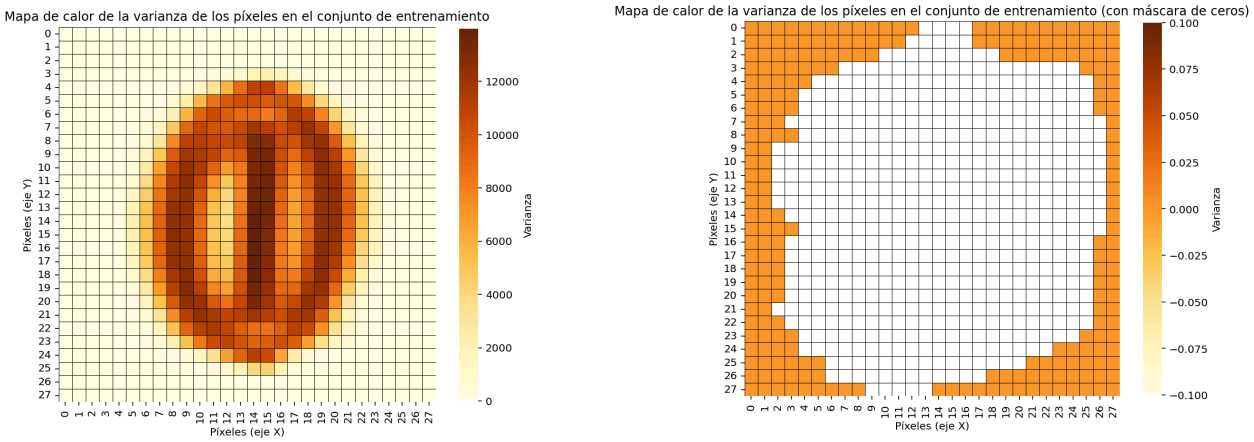
Figuras 14, 15 y 16: Varianzas internas de los dígitos 0, 5 y 9

Así, podemos observar que los atributos de varianza alta están concentrados alrededor de la “forma” que esperamos que tome cada uno de estos dígitos, por lo que podemos afirmar que la mayoría de ellos retienen la forma general deseada. Sin embargo, como notamos anteriormente, existe una gran cantidad de outliers, que nuestro algoritmo de clasificación tendrá que poder distinguir correctamente.

2.2 - Clasificación Binaria:

Comenzando con nuestros experimentos, separamos el data frame en un subconjunto que solo contiene el conjunto de imágenes correspondientes al 1 y al 0, contando con 2990 muestras de cada uno, lo que indica que esta partición ésta efectivamente balanceada. Luego, decidimos dividir el conjunto en entrenamiento y prueba (80% entrenamiento, 20% prueba) para finalmente comenzar a evaluar distintos modelos de KNN, comparandolos segun 4 métricas clave vistas en clase: exactitud, precision, recall y F1.

A modo de experimento, buscamos evaluar distintos modelos de KNN con K entre 1 y 20 para distintos números de atributos. Sin embargo, resulta necesario establecer un criterio de decisión para saber que atributos vamos a elegir. Continuando la misma idea que encontramos en nuestro análisis exploratorio de datos, vamos a observar la varianza para el dataset que solo contiene imágenes de los dígitos 0 y 1.



Figuras 17 y 18: Mapa de calor de varianza del dataset de dígitos 0 y 1 y su mascara de ceros

Podemos observar que tenemos una cantidad acotada de pixeles que contienen información “de valor”, y contamos con 230 píxeles con varianza 0, es decir que siempre se mantienen con el mismo valor (en particular el valor 0), haciendo que no sean relevantes a la hora de determinar a qué digito pertenece la imagen. Así, como criterio de decisión, tomaremos los atributos de mayor varianza para hacer nuestros modelos KNN. A continuación veremos si esto resulta en mejores valores en las 4 métricas mencionadas anteriormente. La comparación seria entre los 3 atributos de mayor varianza (índices 435, 491, 463) el “top 50-52” (índices 357, 548, 402) y el “top 100-102” (índices 523, 238, 629) y comparando a su vez entre train y test. Así, entrenamos distintos modelos KNN utilizando solo estos 3 atributos.

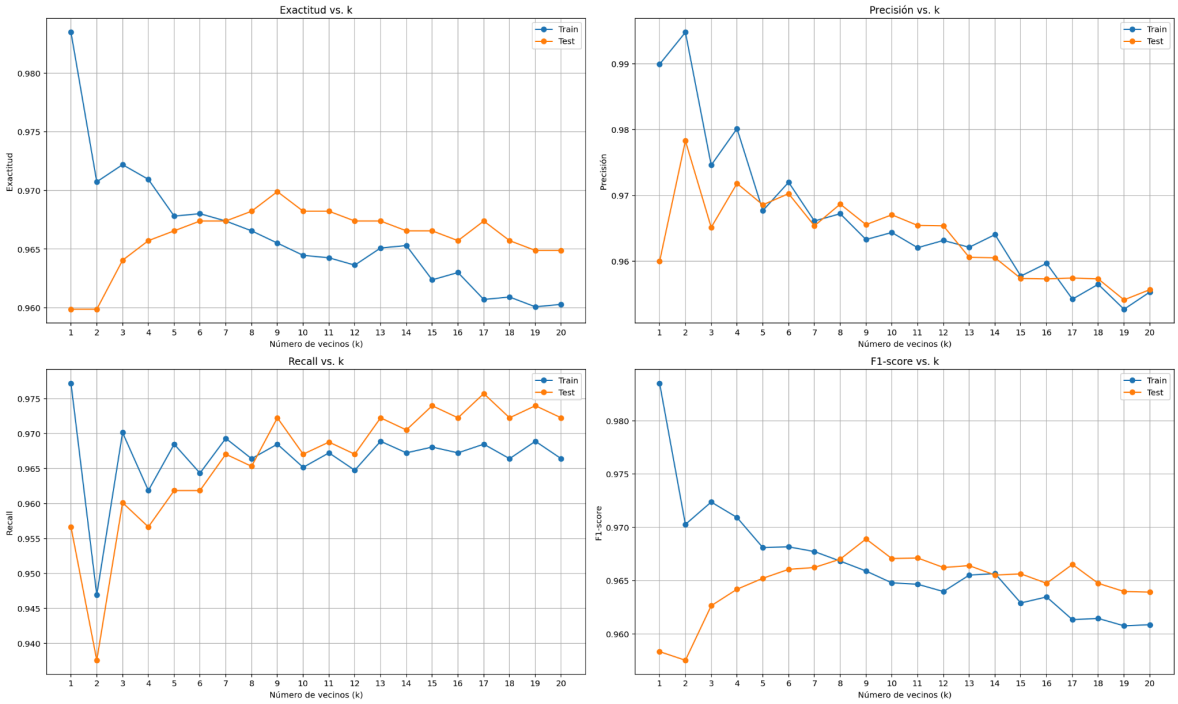


Figura 19: Métricas de modelos KNN sobre los atributos 435, 491, 463 (top 1-3 según varianza)

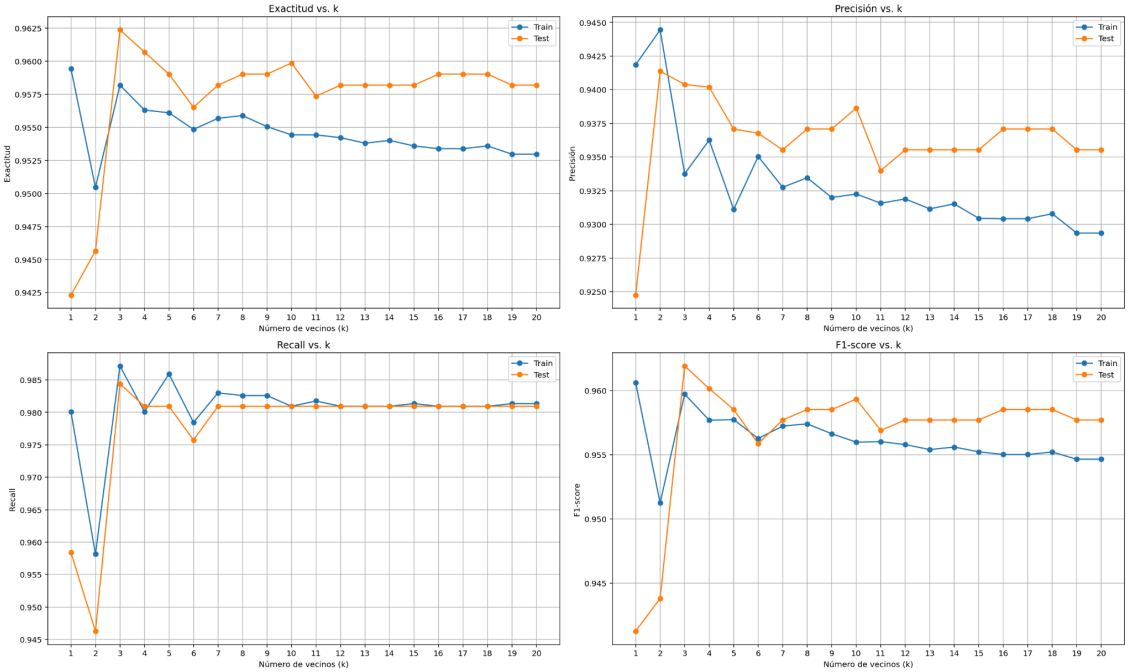


Figura 20: Métricas de modelos KNN sobre los atributos 357, 548, 402 (top 50-52 según varianza)

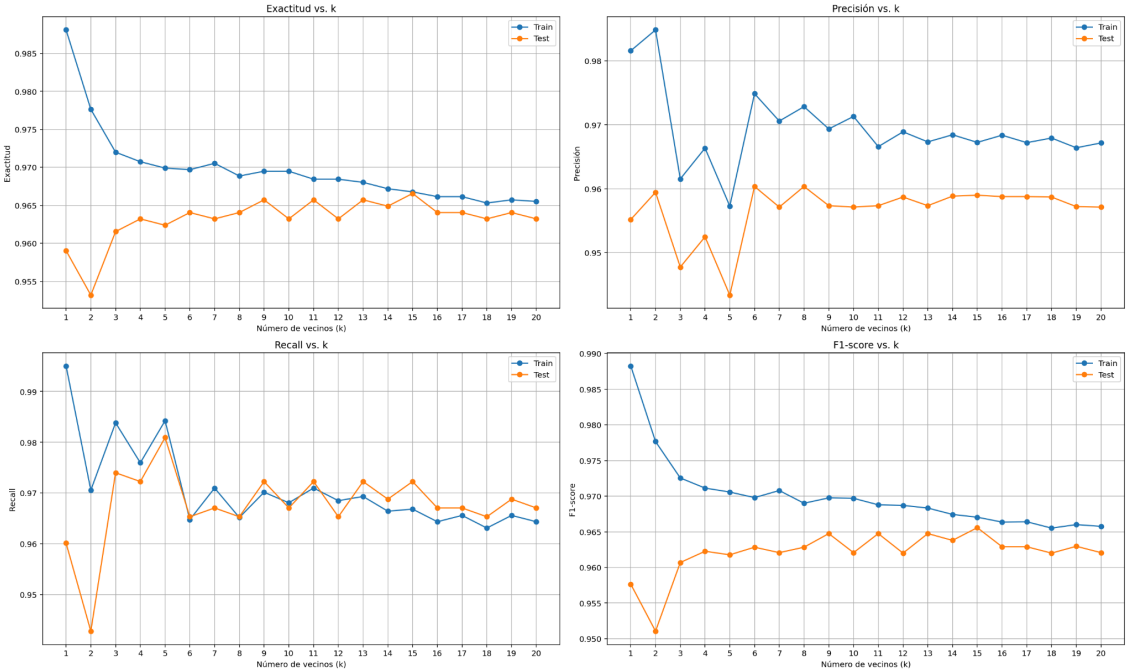


Figura 21: Métricas de modelos KNN sobre los atributos 523, 238, 629 (top 100-102 según varianza)

Vamos a analizar lo visto en cada una de nuestras figuras:

- Figura 19: Se puede observar que K = 9 tiene valores balanceados entre las 4 métricas elegidas. También notamos que la única disparidad significativa entre los resultados en train y test ocurre en cuanto al F1 score.
- Figura 20: Podemos ver que K = 3 tiene los mejores valores para las 4 métricas elegidas, tanto en train como test.
- Figura 21: En este caso, no existe un mejor valor unificado para nuestras métricas, pero podemos tomar K = 6 como una referencia balanceada.

Sin embargo, en las 3 figuras 19-21 podemos observar cómo, a medida que se reduce la varianza de los atributos que elegimos, se reducen los valores de las 4 métricas, que buscamos maximizar, por lo que vemos que nuestra intuición de considerar los atributos de mayor varianza fue una buena decisión. Por último, si tomamos todos los atributos con varianza distinta de 0 (784 - 230 = 534 píxeles de mayor varianza), obtenemos el siguiente gráfico:

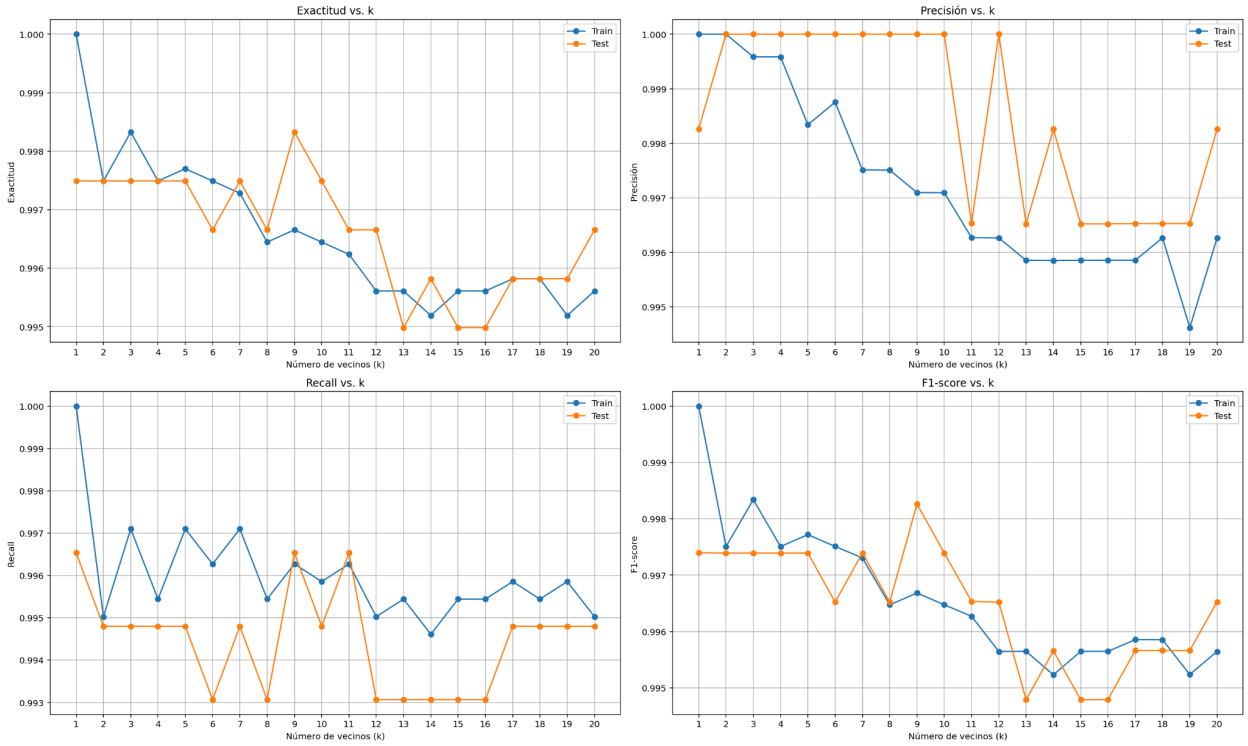


Figura 22: Métricas de modelos KNN sobre todos los atributos con varianza distinta de cero.

Como podemos observar en el gráfico, el mejor valor es el de K = 9, donde se alcanzan los máximos de todas las métricas, y nuestro algoritmo tiene una mucho mejor performance en relación a las mismas. También notamos que aumentar el número de atributos seleccionados no tiene ningún tipo de impacto en los valores de las métricas, pues estamos eligiendo todos los atributos (pixeles) que contienen información relevante a nuestra clasificación.

Por otro lado, repetimos un análisis similar a este, solo que en vez de variar el número de vecinos que tomamos en consideración (k) variamos el número de atributos seleccionados,

entre 1 y 20 de los que más varianza tienen. Así, fijamos en K = 9, pues es el que mejor resultados obtuvo para todos los píxeles con varianza y obtenemos el siguiente gráfico:

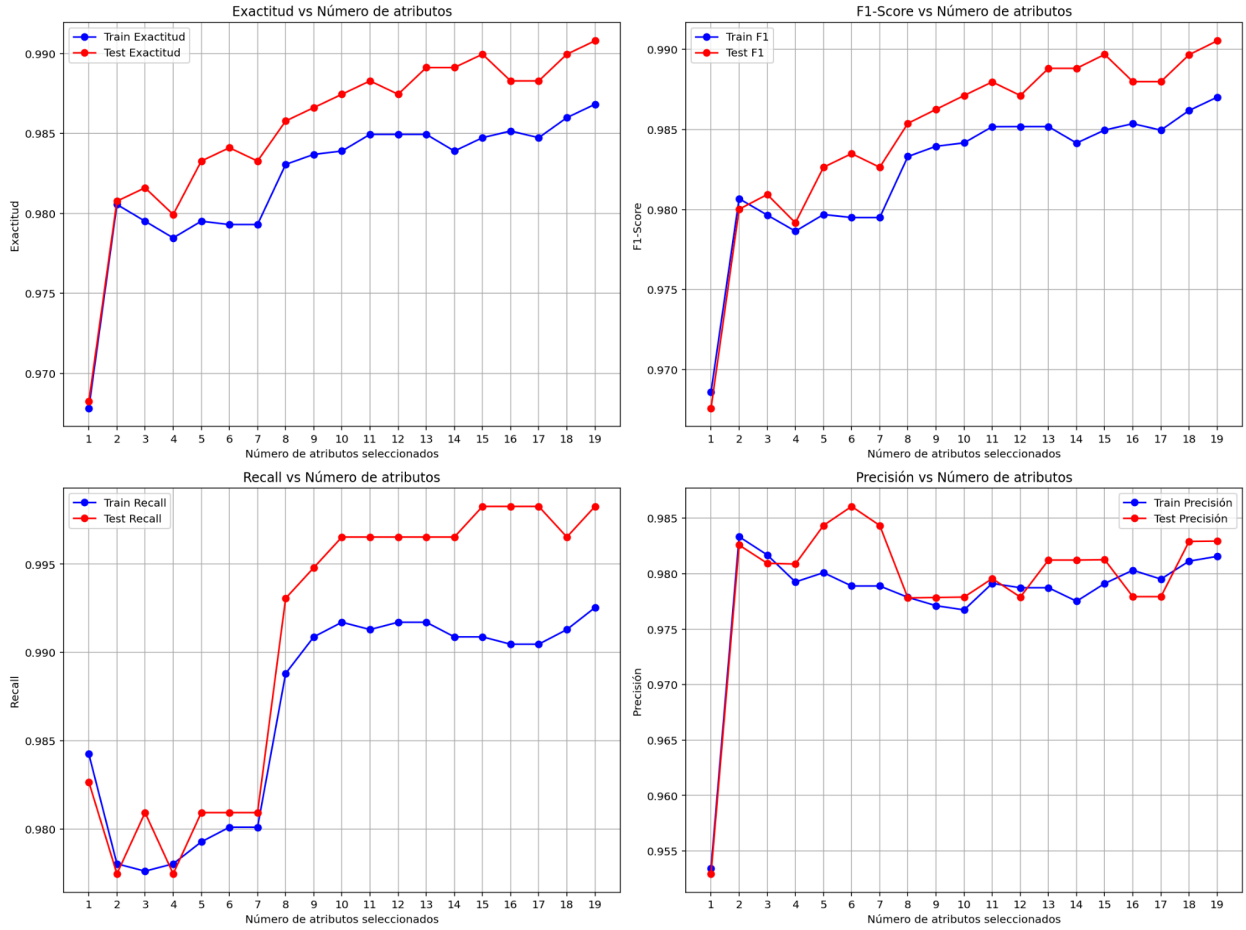


Figura 23: Métricas de modelo KNN con K = 9 tomando distintas cantidades de atributos.

Como podemos observar en la figura 23, la tendencia general es que las 4 métricas tienden a mejorar a medida que aumentamos el número de atributos seleccionados, lo cual coincide con nuestra idea que a medida que seleccionamos mas pixeles entre los que tienen mas varianza, nuestro algoritmo de clasificación mejora. Un caso que contradice esta regla es el de precisión, donde tanto para train como test no observamos una tendencia de crecimiento o decrecimiento de precisión, pues se mantiene en la misma vecindad.

2.3 - Clasificación Multiclase:

Continuamos con una clasificación multiclase, donde se tienen en cuenta los 10 digitos que están en el dataset, y buscamos poder distinguirlos. Para llevar a cabo el entrenamiento y evaluación del modelo de árbol de decisión, se diseñó un proceso que permitió explorar el rendimiento del modelo bajo diferentes configuraciones de hiperparámetros, asegurando además una evaluación robusta y bien fundamentada. Este proceso constó de cuatro etapas:

1) División del conjunto de datos :

Como primer paso, el conjunto de datos completo fue dividido en dos subconjuntos: el conjunto de desarrollo, que representa el 80% del total de los datos, y el conjunto de validación, que corresponde al 20% restante. La intención de esta división fue contar con un conjunto de datos para entrenar y ajustar los modelos (conjunto de desarrollo) y, por otro lado, reservar un subconjunto que no se utilizará durante el entrenamiento (conjunto de validación) para evaluar la capacidad de generalización del modelo sobre datos no vistos. Este conjunto de validación se utilizó en el último paso, cuando se evaluó el rendimiento final del modelo con los mejores hiperparámetros.

2) Ajuste del modelo de árbol de decisión con diferentes profundidades:

En esta segunda etapa, se exploraron diferentes configuraciones de la profundidad del árbol de decisión, variándola de 1 a 10. Para cada valor de profundidad, se entrenó un modelo de árbol de decisión utilizando el conjunto de desarrollo y, posteriormente, se evaluó su exactitud (accuracy) sobre el mismo conjunto de datos.

Figura 24: Exactitud en entrenamiento y prueba en función de la profundidad del árbol

Al representar gráficamente los resultados obtenidos, se observó que la exactitud tanto en el conjunto de entrenamiento como en el de prueba aumentaba conforme incrementaba la profundidad del árbol. Esto indicó que, a medida que el modelo ganaba en complejidad (con mayores profundidades), su rendimiento mejoraba al captar patrones más complejos en los datos. Este comportamiento fue visualizado en la *Figura 24*, la cual muestra cómo la exactitud en entrenamiento y prueba varía en función de la profundidad del árbol, confirmando la tendencia de mejora con árboles más profundos.

3) Validación cruzada con k-folding:

Para evaluar de manera más robusta el rendimiento del árbol de decisión con diferentes profundidades y criterios (Gini y Entropy), se empleó la técnica de validación cruzada con 5 pliegues (k=5). El conjunto de desarrollo fue dividido en 5 partes aproximadamente iguales. En cada iteración del proceso de validación cruzada, una de estas partes fue seleccionada como conjunto de prueba mientras que las otras cuatro se usaron para entrenar el modelo, repitiendo el proceso hasta que cada una de las 5 partes fue utilizada como conjunto de prueba una vez. Esto permitió una evaluación exhaustiva del rendimiento del modelo, utilizando todo el conjunto de desarrollo en diferentes combinaciones para entrenamiento y prueba, y reduciendo así la varianza en la evaluación.

Se exploraron dos criterios de división: “Gini” y ”Entropy”, y se analizó el rendimiento del modelo utilizando profundidades entre 1 y 10. A continuación, se presentan los resultados de la validación cruzada en términos de exactitud media para cada combinación de profundidad y criterio.

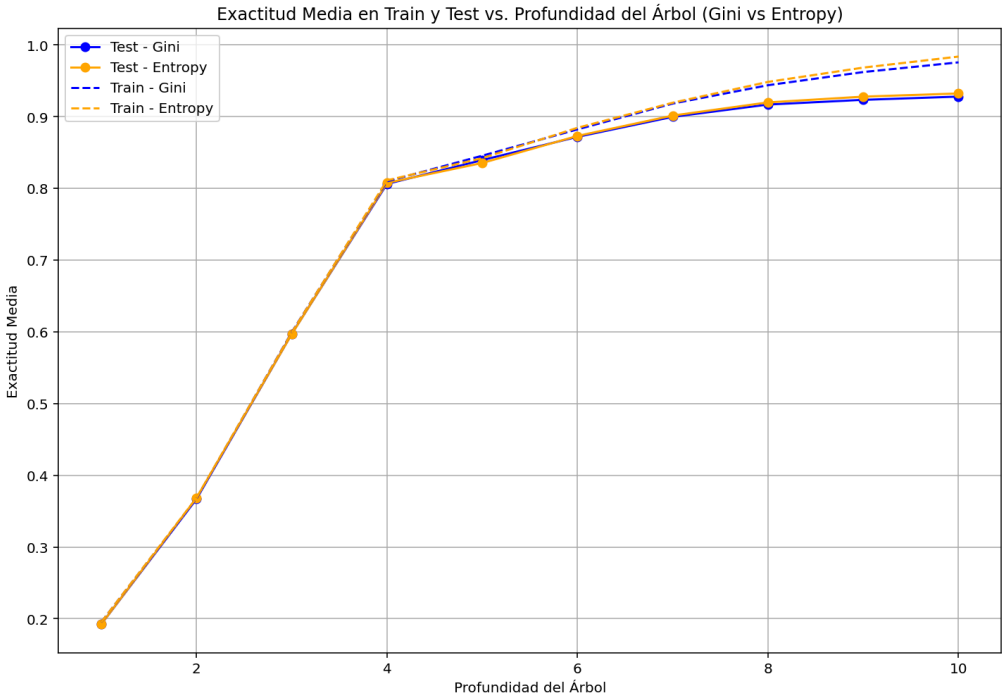


Figura 25: Gráfico de los resultados de la validación cruzada, que presenta la exactitud media en los conjuntos de entrenamiento y prueba para cada profundidad explorada y los dos criterios de división (Gini y Entropy).

De los resultados presentados, se observó que la profundidad 10 produjo la mayor exactitud media en los cinco pliegues para ambos criterios, Gini y Entropy, posicionándose como la mejor configuración para el modelo. Sin embargo, al observar las curvas de exactitud en el conjunto de entrenamiento, se nota que el criterio Gini mostró una exactitud ligeramente más alta en las primeras profundidades, mientras que Entropy superó a Gini a partir de una

profundidad de 6, alcanzando su punto máximo en la profundidad 10. Esta diferencia en el comportamiento del entrenamiento no afectó significativamente la selección final, ya que ambos criterios alcanzaron una alta exactitud en la prueba, pero se seleccionó el modelo con profundidad 10 y criterio Entropy, debido a su mejor rendimiento en el conjunto de prueba.

4) Entrenamiento del modelo seleccionado y evaluación en el conjunto held-out:
En esta última etapa, el modelo de árbol de decisión con profundidad 10 y el criterio Entropy fue entrenado en su totalidad utilizando el conjunto de desarrollo (X_dev, y_dev). Posteriormente, este modelo final fue evaluado en el conjunto de validación (held-out), el cual se había reservado específicamente para este propósito y no fue empleado en ningún paso anterior. La evaluación en el conjunto de validación permitió verificar la capacidad de generalización del modelo sobre datos no vistos.

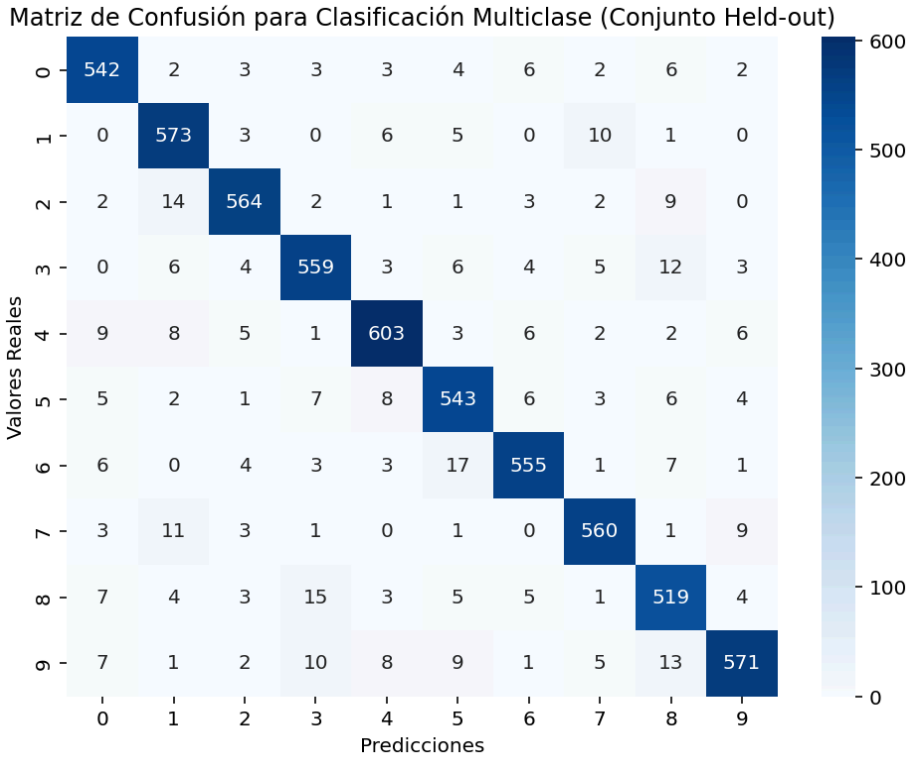


Figura 26: Matriz de confusion

La *Figura 26* presenta la matriz de confusión del modelo final, la cual muestra el desempeño del modelo en la clasificación de cada clase. En esta matriz de 10x10, cada fila representa una clase real y cada columna una clase predicha. Las celdas en la diagonal principal muestran las instancias correctamente clasificadas, mientras que las celdas fuera de la diagonal representan las instancias mal clasificadas, donde se puede ver qué clases tienden a confundirse entre sí. Este análisis a nivel de clase permite identificar si el modelo presenta más errores en ciertas clases específicas, lo cual podría dar pie a ajustes adicionales en futuras iteraciones.

Análisis de las Clases Más Confundidas:

Al examinar la matriz de confusión, se identificaron las clases que más se confundieron entre sí. Las 5 combinaciones de clases más confundidas fueron:

- Clases 6 y 5: Se confundieron 17 veces.
- Clases 8 y 3: Se confundieron 15 veces.
- Clases 2 y 1: Se confundieron 14 veces.
- Clases 9 y 8: Se confundieron 13 veces.
- Clases 3 y 8: Se confundieron 12 veces.

Este análisis revela que el modelo tiene más dificultades para diferenciar entre ciertas clases, especialmente entre las clases 6 con la 5, las cuales fueron las más confundidas. Estas instancias mal clasificadas podrían indicar áreas en las que el modelo podría beneficiarse de ajustes adicionales, como mejorar el preprocesamiento de los datos, ajustar la profundidad del árbol o explorar otros enfoques de modelado para mejorar la precisión en estas clases específicas.

Finalmente, se calculó la exactitud del modelo sobre el conjunto de validación, obteniendo un valor de 0.9346, lo cual indica un buen nivel de generalización del modelo sobre datos no vistos. Esta métrica proporciona una medida concluyente del rendimiento general del modelo final y su capacidad de adaptación a datos nuevos.

3 -Conclusiones:

Tras realizar un analisis exploratorio de los datos del dataset TMNIST, realizamos experimentos sobre clasificaciones binarias y multiclase, con el objetivo de lograr entrenar algoritmos capaces de distinguir las imagenes correspondientes a distintos digitos. Durante este analisis, logramos determinar que la varianza era una buena metrica para distinguir atributos relevantes.

Asi, para la clasificacion binaria entre los digitos 0 y 1 logramos determinar las selecciones mas optimas de atributos para maximizar cuatro metricas: exactitud, precision, recall y F1, tomando los elementos de mayor varianza. Luego analizamos la relacion entre K y estas 4 metricas, determinando asi los mejores valores de K para maximizar las mismas, dependiendo del conjunto de atributos elegido. Finalmente, para concluir nuestra clasificación binaria analizamos la relación entre el número de atributos elegidos y nuestras 4 métricas, observando una tendencia general donde las métricas mejoran a medida que consideramos más atributos.

En cuanto a la clasificación multiclase, logramos identificar la profundidad óptima del árbol de decisión para maximizar la exactitud en la predicción de los 10 dígitos. A través de un proceso de validación cruzada, evaluamos distintas profundidades y concluimos que una profundidad de 10 ofrece el mejor rendimiento tanto con el criterio Gini como con Entropy. Aunque el criterio Gini mostró una ligera ventaja en las primeras profundidades, Entropy superó a Gini a partir de la profundidad 6, alcanzando su punto máximo en la profundidad 10. Posteriormente, evaluamos el modelo final con el criterio Entropy en el conjunto de validación,

obteniendo una exactitud de 0.9346, lo que indica un buen nivel de generalización sobre datos no vistos. Además, se analizó la matriz de confusión para identificar patrones de errores entre clases, destacando las combinaciones de clases más confundidas, lo que sugiere áreas en las que el modelo podría beneficiarse de ajustes adicionales, como mejorar el preprocesamiento de los datos o explorar otras configuraciones del modelo.