Laboratorio de Datos
Trabajo Práctico 2
1er. Cuatrimestre 2025

TRABAJO PRÁCTICO NRO 2

Integrantes: Iván Pérez, Camila Lamblot y Nicole Lamblot
Nombre del grupo: Grupo TP02 - 14
Materia: Laboratorio de Datos
Fecha: 16/06/2025

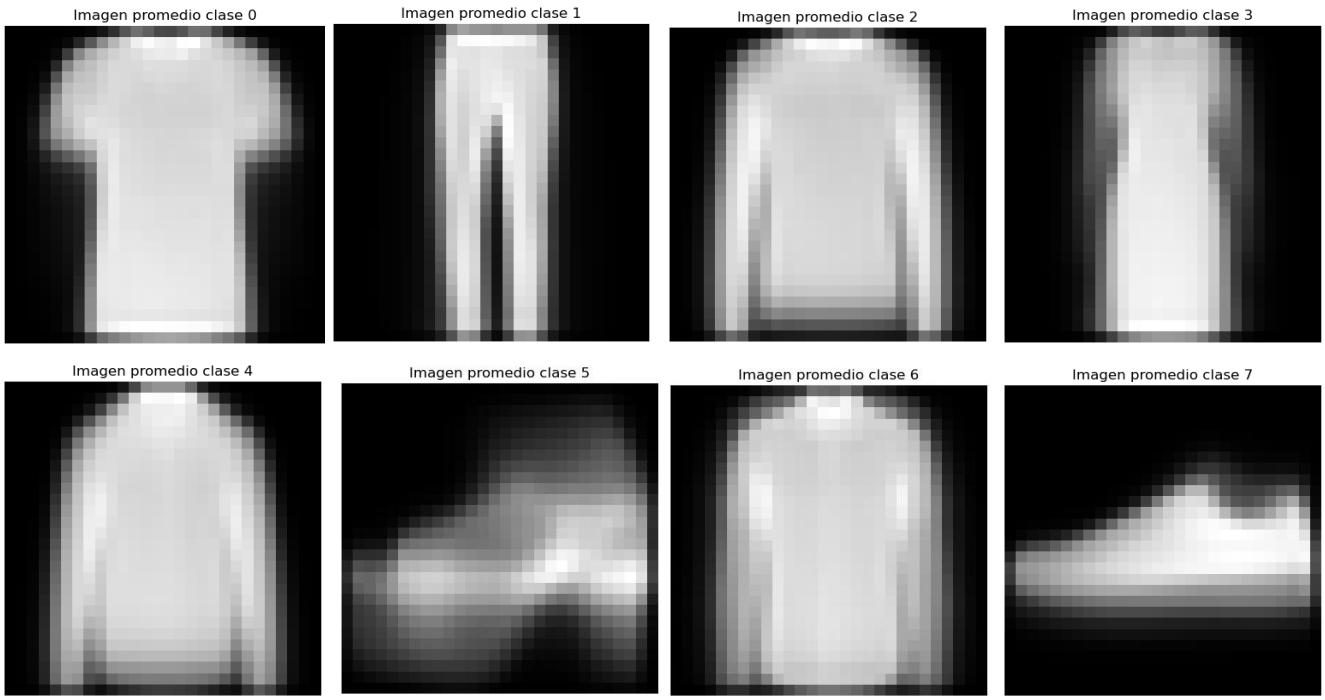
Introducción

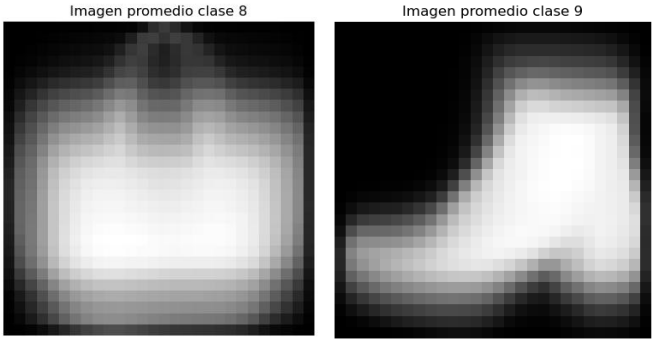
El objetivo de este trabajo es analizar el dataset Fashion-MNIST con el fin de clasificar imágenes de prendas de vestir en distintas categorías. Para ello, realizamos un análisis exploratorio que nos permitió entender mejor la estructura de los datos y seleccionar atributos relevantes. Luego, implementamos modelos de clasificación utilizando kNN y árboles de decisión. A lo largo del trabajo, buscamos no solo ajustar modelos que obtuvieran buenos resultados, sino también comprender cómo influyen factores como la selección de atributos, la cantidad de vecinos o la profundidad del árbol en la performance de los modelos.

Análisis exploratorio

El dataset que se nos presenta corresponde al conocido Fashion-MNIST, compuesto por 70.000 imágenes que representan distintas prendas de vestir. Cada una de estas imágenes pertenece a un gráfico de 28 x 28 píxeles, lo cual se traduce en 784 atributos por prenda. Cada uno representa la intensidad de un píxel en una posición específica. Los valores de los píxeles varían entre los números 0 y 255, e indican cuán claro u oscuro es cada punto de la imagen, siendo 0 negro absoluto y 255 blanco. Las imágenes se encuentran clasificadas en 10 clases distintas, cada una correspondiente a un tipo de prenda; como remeras, pantalones, zapatos, vestidos o bolsos, entre otros. Las clases van de la 0 a la 9, y cada una contiene 7.000 imágenes, de modo que el dataset se encuentra balanceado.

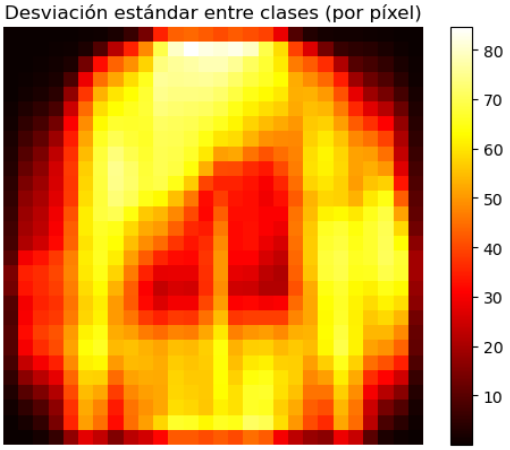
Para el objetivo de nuestro trabajo, nos interesa poder diferenciar los distintos tipos de prendas que se presentan en la base de datos, dado que posteriormente trataremos de predecir a qué clase pertenece cada imagen. Teniendo en cuenta que nuestro dataset no es muy tradicional ya que estamos trabajando con valores de píxeles, vamos a querer encontrar patrones entre clases que nos sean informativos y útiles para poder distinguir cada clase. Por esa razón, decidimos realizar una serie de gráficos que muestran cómo se ve una imágen promedio de cada clase. Para obtenerla, promediamos el valor de cada pixel de todas las imágenes pertenecientes a una misma categoría. El resultado es una imagen representativa que permite observar, de forma general, la forma típica de las prendas de cada clase. De esta forma, podemos ver qué píxeles suelen tener valores similares entre clases y cuáles varían más.





Observando todos los gráficos, lo primero que podemos notar es que los bordes de todas las imágenes son negros. Esto quiere decir que en todas las clases los valores de los píxeles que se encuentran en los bordes de cada imagen son muy bajos y, lo que es más importante, extremadamente similares entre sí. Esto quiere decir que esos atributos no son de utilidad para diferenciar las clases, de modo que podemos descartarlos para nuestro análisis ya que no aportan información relevante.

Asimismo, para profundizar el análisis de la relevancia de los atributos, optamos por evaluar la desviación estándar del valor de cada píxel entre clases. Gracias al cálculo del promedio del valor que toma cada atributo dentro de cada clase, pudimos tomar todos esos valores y calcular cuánto varía cada píxel del promedio entre todas las clases. Esto nos ayuda a comprender cuáles son aquellos atributos que nos sirven para diferenciar las imágenes que pertenecen a distintas clases y cuáles no tanto. Cuanta mayor sea la desviación estándar de un píxel, más relevante se vuelve para nuestro análisis. Mayor desviación representa gran variabilidad entre las distintas clases, mientras que menor desviación indica que dicho atributo tiende a rondar los mismos valores para las distintas clases, lo cual no nos es útil para diferenciarlas. A continuación, podemos observar una imágen que muestra cómo varía cada píxel respecto al valor promedio de todas las clases:



Este gráfico es un Heatmap que muestra claramente qué regiones suelen ser más semejantes entre las imágenes y cuáles no. Los colores más claros representan una mayor desviación estándar, mientras que los más oscuros indican que los píxeles de esa zona suelen presentar los mismos valores en todas las clases. Vemos que los bordes son negros o tienen tonos muy oscuros, de modo que confirmamos lo que habíamos dicho antes: no son relevantes para nuestro análisis. En cambio, la región central presenta tonos rojos y amarillos, de modo que hay mayor desviación estándar. Especialmente la zona central superior tiene colores muy brillantes, casi blancos en algunos casos, que muestran una gran variabilidad de esos píxeles. Por lo tanto, estos atributos probablemente nos serán de gran utilidad para distinguir cada categoría.

A partir de estos análisis, no solo pudimos identificar qué regiones de las imágenes son más informativas, sino que también comenzamos a explorar las similitudes y diferencias entre clases, basándonos en las imágenes promedio obtenidas. Algunas categorías presentan formas muy similares, lo cual podría complicar la tarea de clasificación. Por ejemplo, las clases 2, 4 y 6 tienen una silueta parecida que podría

corresponder a distintos tipos de prendas de la parte superior del cuerpo, como remeras de manga larga, camisas o buzos. Las clases 0 y 3 comparten una estructura semejante a las ya mencionadas, aunque en menor medida, y aparentan ser una remera de manga corta y un vestido respectivamente. Asimismo, los gráficos de las clases 5 y 9 presentan semejanzas, y parecen representar distintos tipos de calzado. Sin embargo, no todas las clases son tan similares: la imagen de la clase 2 (que a simple vista podemos notar que corresponde a un pantalón) no se asemeja a ninguna de las otras clases. Lo mismo pasa con la clase 8, que aparenta ser un bolso.

De esta forma, podemos clasificar a las clases según su semejanza entre ellas: las clases 0, 2, 3, 4 y 6 corresponden a prendas de la parte superior del cuerpo; la 5, 7 y 9 a zapatos; la 1 a pantalones y la 8 a bolsos. Así, se ve que hay una marcada diferencia entre estos subconjuntos, y a la hora de aplicar un modelo que clasifique por sí solo a las imágenes, es esperable que, si comete errores, éstos se produzcan más frecuentemente dentro de estos subconjuntos que entre ellos.

Viendo las imágenes promedio de cada clase, notamos que hay algunas que presentan una silueta más nítida que otras. Por ejemplo, la clase 7 se ve que corresponde a las zapatillas, ya que su forma es claramente reconocible en el gráfico. Sin embargo, identificar el objeto representado en la imagen de la clase 8 puede ser un poco más difícil. No muestra una forma clara, sino que presenta una gran variedad de tonos grises distribuidos en casi toda la imagen, con tonos un poco más claros en el centro de ésta. Aunque podríamos suponer que se trata de un bolso, la imagen no lo deja ver con claridad. Esta falta de nitidez puede deberse a que las imágenes de la clase 8 no son tan parecidas entre sí. De esta forma, al no tener tantos píxeles en común, la imagen promedio de la clase va a ser más difusa, representando una forma general que no muestra con precisión la silueta de cada imagen.

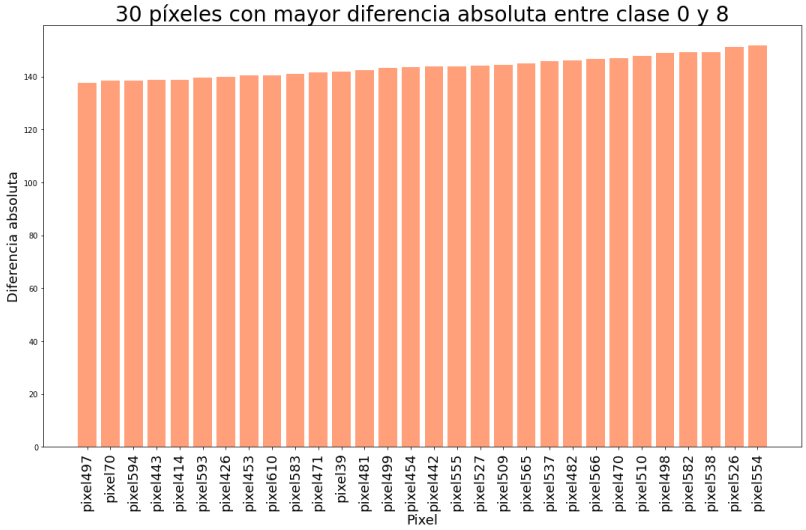
En conclusión, a través de un análisis visual y estadístico de los atributos del dataset, pudimos identificar regiones informativas y redundantes para nuestro objetivo, observar qué clases eran más semejantes entre sí, y detectar categorías con mayor o menor variabilidad interna. Todo este conocimiento es clave para las siguientes etapas del trabajo, en las que necesitaremos de esta información para tomar decisiones que nos lleven a aplicar el modelo de clasificación más adecuado.

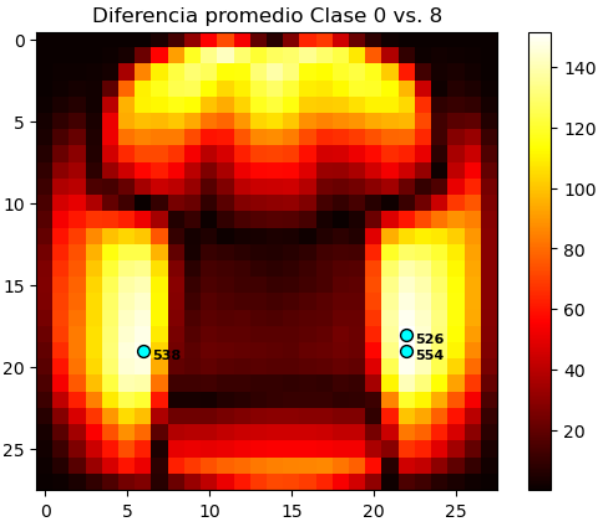
Clasificación binaria

Sabiendo que una imagen pertenece a la clase 0 o a la 8… ¿Podemos predecir a cuál de las dos? Esto es lo que nos propusimos hacer en esta parte del trabajo. Para ello, nos basamos en la clasificación kNN. Construimos un modelo kNN, que, a partir de una cantidad determinada de vecinos, separando nuestro conjunto de datos en datos para training y para testing, pueda predecir a qué clase, (específicamente si a la 0 o a la 8) pertenece una imágen.

Primero comenzamos construyendo un nuevo dataset con las tuplas que correspondían a la clase 0 o a la 8 del original y evaluamos que, efectivamente, está balanceado, habiendo 7000 imágenes en cada una de las dos categorías.

Ahora bien, para realizar este paso, nos interesaba saber cuáles eran los píxeles de mayor varianza para estas clases. Calculando la diferencia absoluta del promedio entre los valores de los píxeles entre ambas, nos quedamos con aquellos 30 que presentaron mayor variabilidad.



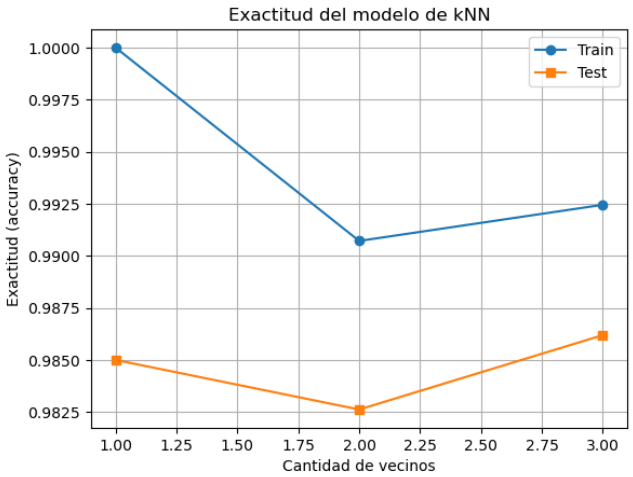


Como podemos ver en los gráficos, aunque todos rondan aproximadamente los mismos valores, los tres píxeles más informativos son el 538, el 526 y el 554, alcanzando una diferencia absoluta de 140.

Luego, nos propusimos ajustar nuestro modelo kNN. Con este objetivo en mente, procedimos a buscar el k (número de vecinos) más robusto para posteriormente evaluar nuestro modelo con distintos subconjuntos y cantidades de atributos, manteniendo k fijo.

Separamos nuestros datos en conjuntos de training y testing de forma balanceada, dedicando el 70% para el training y el 30% restante para el testing; y decidimos limitar a k para que pueda valer de 1 a 3. De esta manera, entrenamos nuestro modelo y lo evaluamos sobre nuestros datos, calculando la exactitud obtenida con el conjunto de datos de training y de testing. Por último, almacenamos las métricas en una matriz, donde la fila corresponde al número de repetición y la columna al k evaluado. Este proceso lo repetimos un total de tres veces.

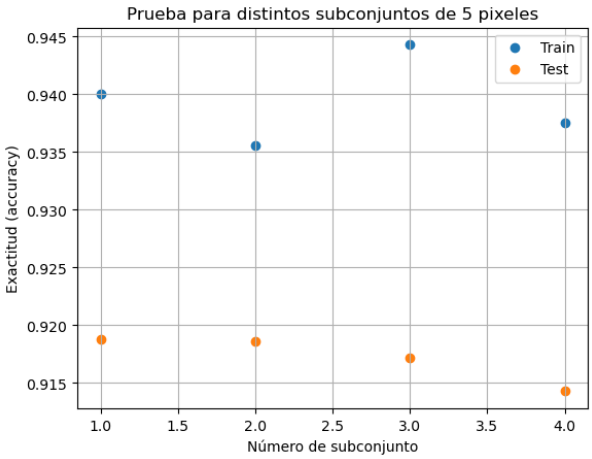
Una vez obtenidos los resultados, construimos un gráfico que nos permita visualizar correctamente cuál es el k más conveniente.



Vemos que cuando k = 3 obtenemos la mayor exactitud en el testing, por lo tanto, establecemos k = 3 para los próximos ajustes.

Ahora, sabiendo que lo más estable es usar el k=3, resta probar distintos subconjuntos para entrenar al modelo. Para esto, tomamos el conjunto de treinta píxeles de mayor varianza que obtuvimos antes y elegimos 4 subconjuntos de 5 atributos al azar. Hacemos esto para poder comparar la exactitud del modelo para distintas combinaciones de atributos, basándonos solamente en el criterio de que pertenecen a los píxeles de mayor variabilidad.

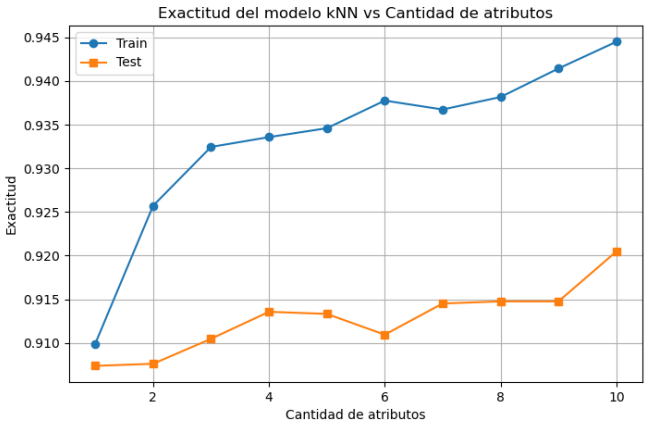
Entonces, repetimos el mismo proceso que antes, con la diferencia de que ahora dejamos a k fijo y lo que varía es el subconjunto de atributos. Además, en vez de renovar los conjuntos de datos de training y de testing en cada repetición, optamos por mantener el mismo en todas las repeticiones con el fin de poder evaluar específicamente la influencia de cada subconjunto en la exactitud del modelo.



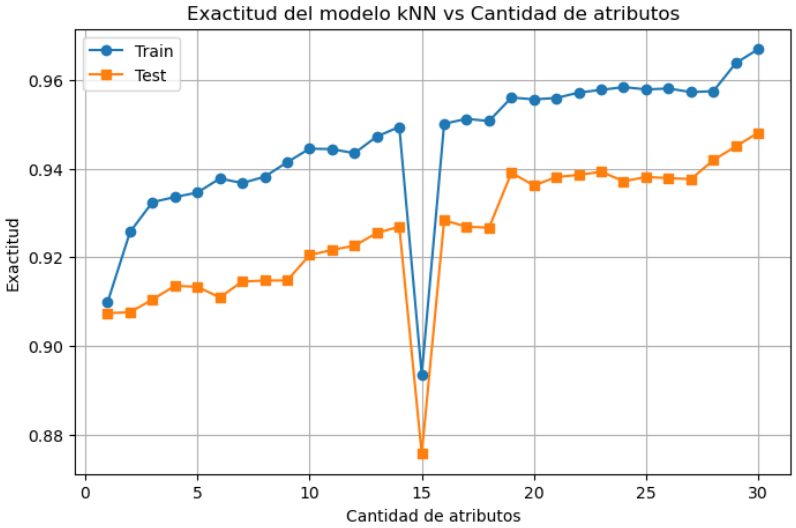
En este gráfico podemos observar que el primer subconjunto es el que produce la mayor exactitud.

A pesar de haber obtenido una buena exactitud siguiendo este método, nos pareció importante ver qué sucedía si nosotros mismos elegíamos los atributos a testear, siguiendo un criterio que nos sea de utilidad. Es por eso que decidimos quedarnos con los 10 píxeles de mayor varianza y formar 10 subconjuntos que contienen de 1 a 10 atributos, respetando el orden de mayor varianza. Decidimos hacer esto puesto que al tomar los píxeles que más varían entre las dos clases, podemos ajustar el modelo y probablemente obtener una mejor clasificación que si lo hiciéramos al azar.

En esta vuelta, no solo modificamos los subconjuntos de atributos, sino también la cantidad. Por lo tanto, volvemos a repetir el mismo procedimiento, dejando fijo únicamente el k y las repeticiones.

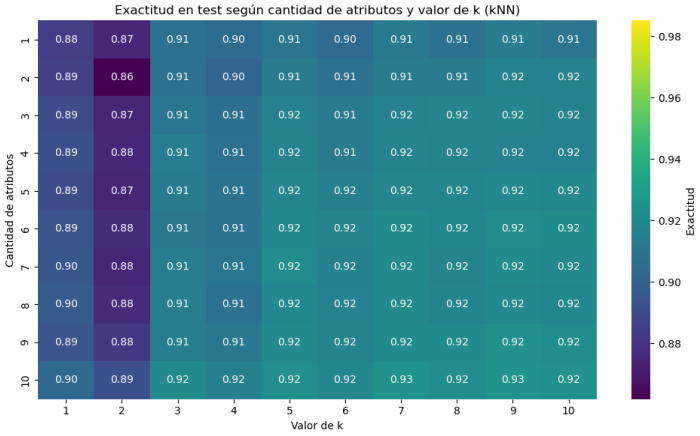


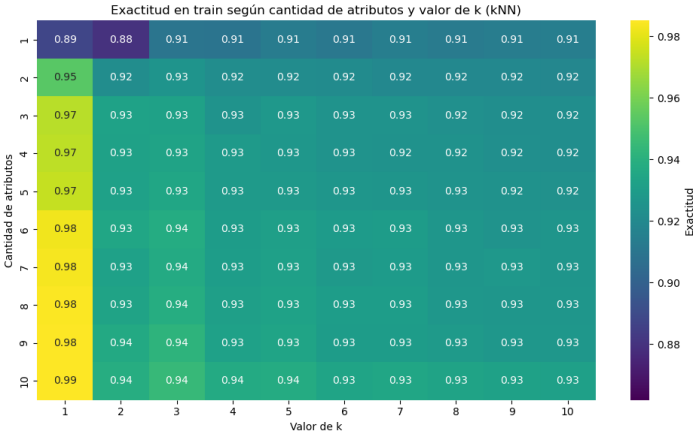
Mirando la visualización, notamos que tanto la exactitud del training como del testing sube a medida que aumenta la cantidad de atributos, lo cual es lógico. Dado que haciéndolo hasta 10 atributos el modelo no pareciera 'estancarse', decidimos probar cómo se vería usando hasta 30 atributos, y obtuvimos lo siguiente:



Como podemos ver, hay una tendencia creciente, aunque con k=15 la exactitud disminuye de forma abrupta. Luego, alrededor de los 19 atributos hay una especie de meseta, es decir, la exactitud se mantiene más o menos constante entre los atributos 19 y 28, por lo que más que eso ya sería overfitting. Por lo tanto, la mejor cantidad de atributos para ajustar el modelo sería 19, pero observando que hay una variación de 0,02 de exactitud entre 10 y 19 atributos conviene utilizar 10 atributos (usar 19 atributos implica una complejidad mucho mayor).

Por último, quisimos analizar cómo varían los resultados de nuestro modelo si además de la cantidad y subconjuntos de atributos variamos el k. Es así como almacenamos en una matriz las exactitudes obtenidas según el valor de k y la cantidad de atributos. Esto lo hicimos para los datos de training y para los de testing. A partir de ello, representamos los resultados en 2 Heatmaps con la misma escala de colores, de manera que podamos visualizarlos y compararlos.





Con respecto al Heatmap de train, vemos que con k = 1 obtenemos valores de exactitud muy altos, tales como 0.99, lo cual sugiere que se produjo un sobreajuste. Asimismo, desde los 3 atributos en adelante, la exactitud en train es bastante estable, (alrededor de 0.93 y 0.94), incluso con valores altos de k.

Por otro lado, aunque es esperable, el modelo consigue mejor exactitud a partir de los datos de training que los de testing. Notemos que la variabilidad de k sobre todo en este último es considerablemente más influyente que la cantidad de atributos: esto se hace evidente en los valores de cada una de las columnas de test, siendo todos iguales o muy similares entre sí. De modo que utilizar más de 4 o 5 atributos no aporta una diferencia significativa.

Los valores de exactitud de test están entre 0.86 y 0.93, mostrando menor varianza, siendo los valores más bajos de k (k=1 y k=2) los que tienen peor desempeño. Así vemos que, efectivamente, se produce un overfitting en train en esos valores. En general, a partir de k=3 los valores mejoran y se estabilizan.

Ahora, recordando cuando probamos cómo variaba la exactitud del modelo al aumentar la cantidad de atributos, dejando k = 3 fijo, vemos que, en general, mientras más atributos usaba, mejor desempeño obteníamos tanto en train como en test. Pero después, al ver los mapas de calor que comparan distintas combinaciones de atributos y valores de k, notamos que a partir de 4 o 5 atributos la exactitud en test ya no mejora mucho.

Entonces, aunque agregar más atributos puede parecer que ayuda (y en algunos casos lo hace), en la práctica no aporta una mejora significativa después de cierto punto. Además, usar demasiados atributos puede llevar a que el modelo se sobreajuste o sea más complejo de lo necesario.

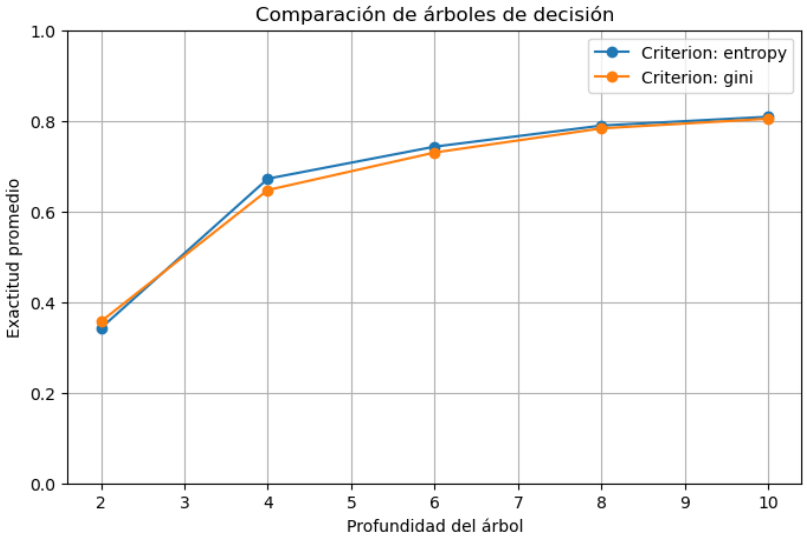
Por eso, lo más razonable sería quedarse con entre 3 y 5 atributos bien elegidos, y usar un valor de k entre 5 y 9, que son los que dieron mejores resultados en general.

Clasificación multiclase

Para esta última parte del trabajo, nos propusimos ajustar un modelo de árbol de decisión utilizando validación cruzada con k-folding que, dada una imagen incógnita, permitiera clasificarla en alguna de las 10 clases pertenecientes al dataset.

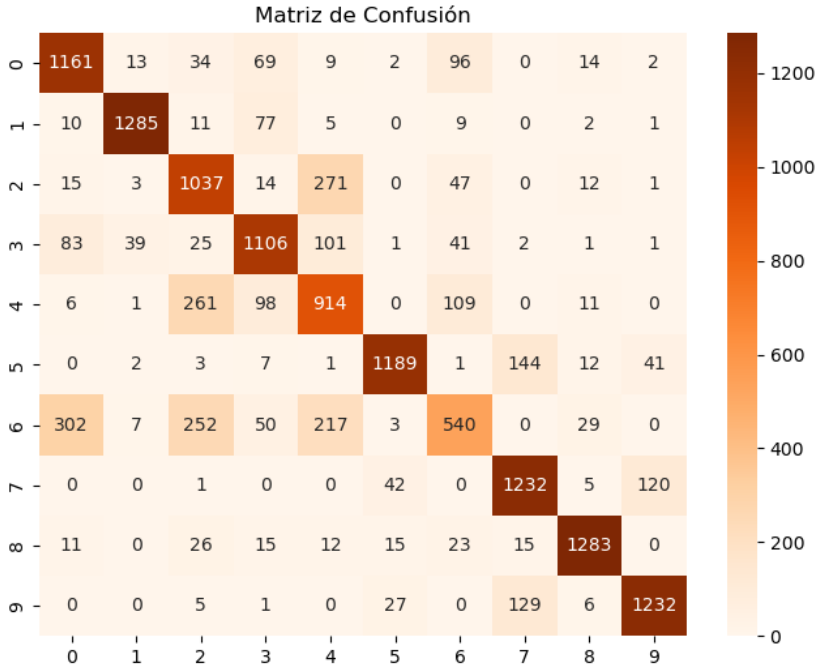
En primer lugar, separamos nuestros datos en dos conjuntos de manera balanceada. Destinamos el 80% de ellos al desarrollo (dev) y el 20% restante a la validación (held-out).

Luego, definimos los hiperparámetros que testearíamos para ajustar el mejor árbol de decisión. Optamos por variar el criterio de elección de atributos en cada nodo, utilizando entropía y gini; y variamos la profundidad del árbol, tomando únicamente los números pares del 1 al 10 para optimizar el tiempo de ejecución.



Creamos este gráfico para visualizar mejor el desempeño de cada modelo según ambos hiperparametros. Como podemos observar, no hay una diferencia significativa entre entropía y gini. Sin embargo, la exactitud mejora a medida que la profundidad del árbol aumenta, estabilizándose a partir de la 8, donde los criterios entropía y gini alcanzan casi la misma exactitud, el primero teniendo tal vez un valor ligeramente mayor. Por esta razón, decidimos ajustar el modelo final utilizando entropía y una profundidad igual a 8 (viendo que que no mejora en demasía la exactitud entre una profundidad de 8 y 10).

Una vez definido y entrenado el modelo final con el total de datos de desarrollo, tomamos el conjunto held-out y predijimos la clase a la que pertenece cada una de sus imágenes. Para evaluar la performance, construimos la siguiente matriz de confusión:



Observamos que la diagonal contiene, por mucho, los valores más altos de la matriz, lo cual es deseable, puesto que cada elemento de la diagonal representa la cantidad de imágenes que fueron clasificadas correctamente según la clase a la que corresponden.

La clase 6 es la que cuenta con el peor desempeño, seguida por la 4 y la 2. Si recordamos los gráficos promedios que construimos en el análisis exploratorio, sabemos que estas tres clases eran las más semejantes entre sí, pues son prendas correspondientes al tren superior de manga larga, de manera que comparten muchos píxeles con valores similares. Por lo tanto, tiene sentido que el modelo tienda a fallar más con estas clases que con el resto.

Por otro lado, la clase 1 es la mejor calificada, siendo ésta la que pertenece a los pantalones. Dado que es la única clase que representa una prenda de la parte inferior del cuerpo, sin tener en cuenta el calzado, es lógico que el modelo haya podido diferenciarla mejor que el resto.

Finalmente, calculamos la exactitud del modelo sobre el conjunto held-out y el resultado fue de 0.7842142857142858. Cabe destacar que es un valor muy similar al que calculamos con el conjunto de desarrollo para el criterio de entropía y profundidad igual a 8.

Conclusiones

En conclusión, si comparamos los modelos que usamos, vimos que el kNN funciona muy bien cuando el problema está bien acotado, como en la clasificación binaria entre las clases 0 y 8. En ese caso, elegir bien los atributos marcó la diferencia: usar pocos píxeles, pero bien seleccionados rindió más que elegir aleatoriamente. También notamos que valores bajos de k sobreajustan, mientras que con k entre 5 y 8 el modelo fue más estable.

En cambio, para la clasificación multiclase, el árbol de decisión fue útil considerando la complejidad. Aunque la exactitud fue un poco menor, logró buenos resultados incluso con clases que eran difíciles de distinguir. Acertó más cuando las clases eran visualmente distintas, y falló donde ya sabíamos que se parecían (como las prendas de manga larga).

En resumen, los dos modelos tienen sus ventajas. kNN depende mucho de cómo elegimos los datos, pero suele ser muy exacto. El árbol es más robusto para problemas grandes, aunque un poco menos preciso. Concluimos que, a partir de un exhaustivo análisis exploratorio y probando las distintas variables que influyen en cada modelo, pudimos tomar decisiones más informadas, y eso tuvo un impacto directo en la performance de los modelos.