Laboratorio de Datos

Trabajo práctico 2

Modelos de Clasificación

Integrantes: Guillermo de la Vega, Matías Naddeo, Francisco Anllo

Docente: Rodrigo Laje

2025

Facultad de Ciencias Exactas y Naturales

Introducción

En este informe, diseñamos un modelo de clasificación para separar y analizar una base de datos que corresponde a un catálogo de prendas. Dicho dataset cuenta con diez vestimentas distintas, y por cada una de ellas, siete mil ejemplares. Para realizar esta tarea, ha sido necesario estudiar varias muestras de cada una a través de fotos monocromáticas en escala de grises. El desafío residía en intentar discernir cómo sería posible separar cada tipo distinto de prenda con solo un número reducido de píxeles. A continuación, detallamos nuestro proceso de investigación, seguido del entrenamiento de nuestro modelo de clasificación y, por último, registrar cuánta precisión predictiva posee.

Análisis exploratorio

Antes de comenzar, una breve descripción de la estructura del dataset. Este cuenta con 786 columnas, de las cuales la primera no tiene nombre y es únicamente de índices, por lo que no la vamos a necesitar. También está la columna “label”, que indica la clase de la prenda. Las otras 784 representan cada píxel: “pixel0”, “pixel1”, ... ”pixel783”, donde el valor indica la tonalidad de gris, 0 para completamente negro y 255 para blanco.

Para tener un panorama de qué píxeles son los más relevantes a la hora de distinguir entre clases, una estrategia que surgió naturalmente fue encontrar la silueta general que dibujan los elementos de cada una de las clases y encontrar patrones comunes que nos ayuden a diferenciarlas.

Primero, quisimos hacernos una idea de cómo se ve el dataset en general, por lo que calculamos un promedio global por cada pixel.

Usamos esta medida de tendencia ya que en este caso los valores están restringidos entre 0 y 255, y además consideramos que la cantidad de datos es demasiado elevada como para verse influenciada por valores extremos.

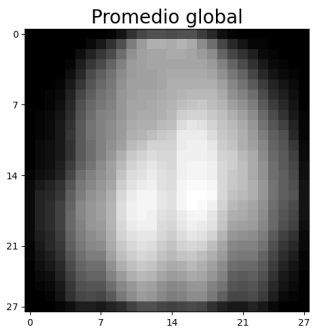


Imagen 1: promedio de todas las clases

Es imperativo aclarar que el análisis explicado en esta sección fue realizado a partir de un recorte del dataset original. Como hemos dicho, contábamos con siete mil ejemplares de cada clase, pero nosotros trabajamos con una muestra de mil ejemplares por clase. Consideramos que era una muestra lo suficientemente amplia para no perder información esencial, y al mismo tiempo optimizar el tiempo y facilitar la visualización. No obstante, a la hora de ejercitar y calificar nuestros modelos, sí utilizamos el dataset original, por supuesto.

Volviendo al análisis, observemos la imagen 1, donde se superponen las imágenes promedio de todas las clases. No parece decir mucho, pero pudimos sacar ciertas conclusiones. Notemos que valores más oscuros indican que las prendas no suelen llegar a esas zonas, mientras que valores relativamente altos, o aunque sea no tan bajos, muestran que hay muchas prendas que cubren ese espacio. Con este análisis, la visualización nos permitió concluir que, como anticipábamos, las esquinas suelen tener valores muy oscuros, cercanos a 0. También vimos que pixeles muy centrales no aportaban tanta información, ya que pareciera que casi todas las prendas cubren esas zonas. Por ende, pensamos que aquellos píxeles no nos serían de utilidad para el propósito de este informe, mientras que los que más nos servirían son los que rodean al centro.

Luego, procedimos a graficar exactamente lo mismo que antes, pero agrupando prendas por clase: Nuevamente, el resultado fue el mismo que se vio luego de superponer cada una de las fotos. De esta forma obtuvimos una figura promedio por clase.

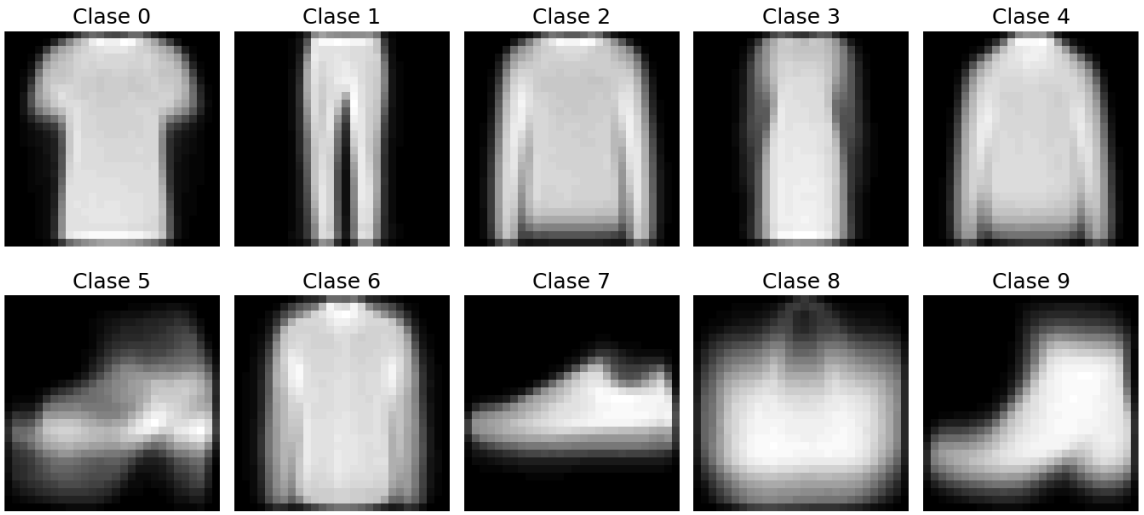


Imagen 2: figura promedio por clase

Cabe mencionar que, dado que sabemos que tenemos la misma proporción de cada una de las clases, podemos asegurar que esta es una descomposición por clase de la imagen 1.

Antes de entrar en diferenciación entre clases, debemos destacar que hay algunas que están más borrosas que otras. Esto evidencia que hay clases que tienen más variación en forma que otras. Por ejemplo, las clases *1*, *2* y *7* muestran una forma más definida, con bordes menos borrosos, lo que nos lleva a entender que las prendas de esta clase suelen ajustarse más a la figura aquí representada. Es notorio también el caso de la clase *5*, que al haber tantos tipos distintos, si bien podemos notar una leve tendencia general, esta aparece más borrosa. Para ver de manera más clara la variación entre los elementos de una misma clase, graficamos un heatmap con la desviación estándar de cada clase píxel por píxel.

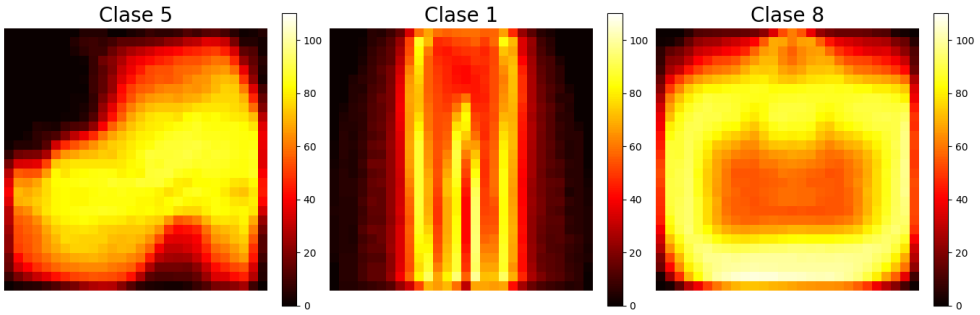
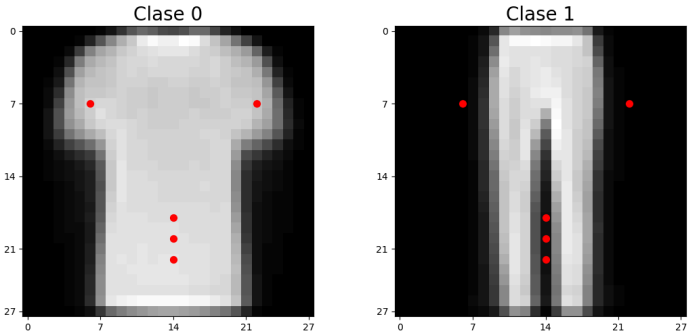


Imagen 3: heatmaps de la varianza de los píxeles de las clases 1, 5 y 8

Ahora podemos ver más claramente que hay clases que son más homogéneas que otras. La clase *5* es el ejemplo más claro de una clase que varía considerablemente tanto en forma como en tamaño. Por otro lado, la clase *1* apenas cambia en su forma, a lo sumo varía un poco más el tamaño. Los elementos de la clase *8*, por otro lado, si bien sucede que a veces tienen manija y a veces no, la principal alteración se presenta en su tamaño, aunque respetando generalmente una forma cuadrada.

Siguiendo con el análisis, nos valemos de los siguientes gráficos. En ellos se pueden observar los patrones generales de cada clase que nos ayudarán a diferenciar algunas de otras. Por ejemplo, para distinguir entre un elemento de la clase *0* y uno de la clase *1*, podríamos mirar las mangas o el espacio entre las piernas del pantalón.



Vemos entonces, que al fijarnos el valor de estos pixeles, resulta claro cómo diferenciar entre un elemento de clase *0* y uno de clase *1*. A esta forma de visualización la llamaremos, a partir de ahora, **método 1.** Otros ejemplos de clases que son distinguibles de manera clara usando este método son la clase *4* con la *7*, la clase *3* con la *9*, la clase *1* con la *2*, y varias más, que mencionaremos más adelante.

Desafortunadamente, este método no fue suficiente para separar todas las clases, pues aquellas que tienen formas más similares entre sí no presentan diferencias notables a primera vista. Un ejemplo son las clases *2*, *4* y *6*. Como sus diferencias son muy sutiles, no es fácil ver algún píxel que las diferencie. Esto nos llevó a tener que encontrar una mejor forma de buscar píxeles que nos permitan clasificar bien entre clases más parecidas.

Para este fin, desarrollamos otro método, fundamentalmente basado en el método 1, que dadas dos clases y sendos promedios de cada pixel, calculamos la diferencia entre dichos promedios. Colores más claros indican promedios más distintos. Así que bastó con obtener aquellos píxeles con valores más altos. Este será llamado **método 2**.

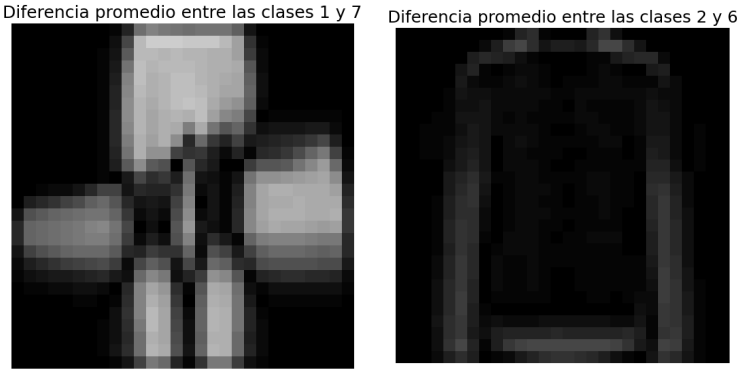


Imagen 4: diferencia del promedio de los píxeles entre las clases 1 y 7, y 2 y 6

Nuestros dos métodos para encontrar píxeles se basan en la idea intuitiva de que dado un píxel, a mayor diferencia de promedio entre clases, mejor es para usarlo en nuestra clasificación. Sin embargo, hasta este punto, hemos asumido que es cierto, en vez de comprobarlo. Por lo tanto, nos propusimos corroborar que esta afirmación es verdadera antes de seguir avanzando. Para esto, dadas dos clases arbitrarias, graficamos un punto por cada pixel, formando un *scatterplot.* En el eje *x* está la diferencia promedio por clase, mientras que en el eje *y* se halla la *exactitud* del algoritmo más sencillo y natural que surge. Tomamos como ejemplo las clases *2* y *6*, que ya hemos dicho que no presentan diferencias notorias.

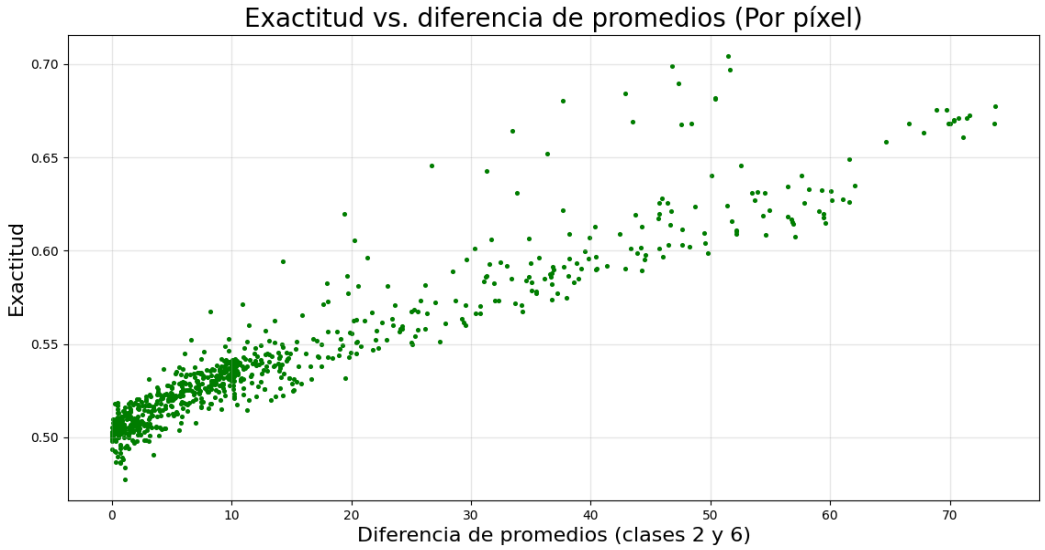


Gráfico 1: relación entre las diferencias de promedios de las clases 2 y 6, y la exactitud del modelo

Viendo estos resultados, observamos que efectivamente, mayor diferencia en el promedio resulta en un aumento de la exactitud del algoritmo (por lo menos con este método), qué era lo que buscábamos. Aunque es verdad que no graficamos las relaciones de todas las clases entre sí, el principio es el mismo. Por todas estas razones, consideramos que era buena estrategia usar los píxeles con mayor diferencia promedio para clasificar, y seguimos adelante.

Con estos dos métodos ya mencionados, proseguimos entonces a intentar distinguir cada clase. Creamos nuestro propio "árbol de decisión" con el fin de aislar cada clase de las demás. A continuación, se detalla nuestro tren de pensamiento, y con qué método obtuvimos los píxeles correspondientes a la clasificación.

Separamos en dos grupos grandes con los píxeles *40*, *68*, *96* y *126,* obtenidos a través del método 1 y verificados también con el método 2:

- **GRUPO A** = clases *0, 1, 2, 3, 4 y 6.* Los dividimos por el pixel *454* (deducido a partir del método 1) y conseguimos:
  - **GRUPO A1** = clases *0,1 y 3*. Para terminar de separar estas clases, usamos los píxeles *173, 145 y 201*, que realizan esta separación:
    - Clase *0*
    - Clases *1 y 3*. Para diferenciarlas, usamos el pixel *574*.
    - Todos los píxeles para separar a este subgrupo fueron conseguidos a través del método 1
  - **GRUPO A2** = clases *2, 4 y 6*. Aquí la clasificación es más compleja, pues no encontramos píxeles concretos que las separen adecuadamente. Sin embargo, pudimos discernir un set de píxeles que las dividían de forma **binaria.**
    - Usamos el píxel *63* para separar entre las clases *2 y 4*.
    - Entre las clases *2* y *6*, dividen bien los píxeles *733* y *38*.
    - Para separar entre las clases *4 y 6*, lo mejor que encontramos fueron los píxeles *427* y *399*.
    - ¡Uniendo estos sets de píxeles -obtenidos todos a través del método 2-, podemos asumir que las clases terminaron bien distinguidas!

- **GRUPO B** = clases *5, 7, 8 y 9.* A su vez, separamos estas clases utilizando los píxeles *579* y *607*, obtenidos a través del método 2, de la siguiente forma:
  - **GRUPO B1**: clases *8 y 9*. A estas clases las separamos según los píxeles *340* y *312*, adquiridos mediante el método 1.
  - **GRUPO B2**: clases *5 y 7*. Para separarlas, usamos los píxeles *380* y *381*, que a su vez recogimos gracias al método 2.

Adicionalmente, aunque ya habíamos encontrado suficientes píxeles para distinguir a cada clase -al menos en cierta proporción razonable-, queríamos separar más precisamente a las clases *0* y *8*. Para ello, utilizando el método 2 mencionado anteriormente, descubrimos una serie de píxeles que nos ayudarían para este propósito: el *70*, *526* y *554*.

Finalmente, dimos por concluido nuestro análisis exploratorio. Luego de un arduo proceso de búsqueda de los píxeles que funcionan mejor como atributos de nuestro modelo de clasificación, terminamos con un conjunto de 23 píxeles, una cantidad que consideramos razonable, dado que había 784 píxeles posibles para elegir.

Modelo de clasificación

En esta sección, ajustaremos un modelo de clasificación Knn para intentar separar las prendas de clase 0 de las de clase 8. En primer lugar, sabíamos que hay siete mil ejemplares de cada prenda, por lo que, afortunadamente, la cantidad de muestras en cada clase era la misma. Esto facilitó el proceso, ya que la proporción de elementos no está desbalanceada, y pudimos separar los conjuntos de *train* y *test* sin dificultades.

Ahora, para definir los atributos que utilizamos, nos remitimos al análisis exploratorio que habíamos realizado previamente. Habíamos visto que los píxeles *40, 68* y *96* dividían a las clases en dos subgrupos, y la clase 8 terminaba en otro grupo que la clase 0. Por lo tanto, consideramos que este subconjunto de píxeles era un buen candidato para entrenar a nuestro modelo. Repetimos el proceso de entrenamiento y testeo para cada valor de vecinos (*k*) entre uno y veinte. Vale la pena mencionar que solo realizamos este proceso una única vez para cada valor de *k,* pues como la semilla está fija, los conjuntos de entrenamiento y testeo son los mismos siempre; es decir, no habría variación entre ejecuciones para un mismo valor de *k*. Por lo tanto, bastó con calcular la precisión y volcar la información en un gráfico.

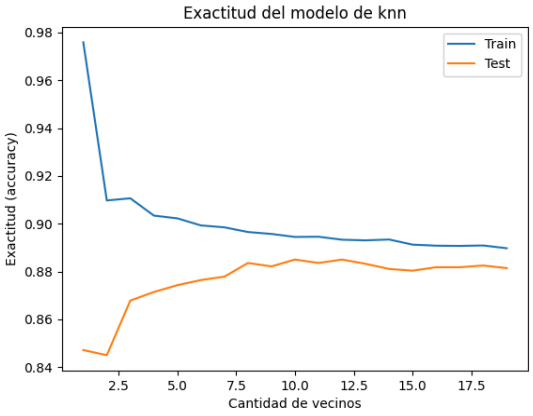


Gráfico 2: precisión del modelo KNN en función de k con los píxeles 40, 68 y 96

Podemos observar que se obtiene la mayor precisión cuando *k = 12*. Específicamente, conseguimos los siguientes puntajes en cada medida de performance con esa cantidad de vecinos:

- *Accuracy*: 0.885
- *Precision*: 0.8778735632183908
- *Recall*: 0.8893740902474527
- *F1 score*: 0.8835864063629791

No son valores bajos en absoluto. Sin embargo, habíamos descubierto, durante el análisis exploratorio, un set de píxeles que funcionaban de manera excepcional para separar estas dos clases: los píxeles *70, 126, 526* y *554*. Por lo tanto, repetimos el mismo proceso y graficamos la información:

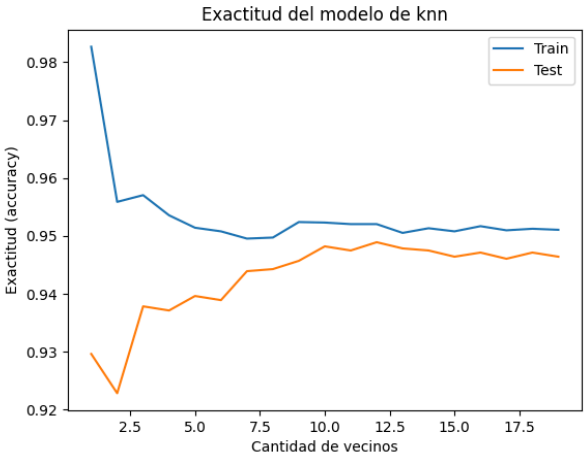


Gráfico 3: precisión del modelo KNN en función de k con los píxeles 70, 126 y 526 y 554

¡Efectivamente, este subconjunto de píxeles es mejor a la hora de separar entre estas clases! Podemos ver que, cuando *k = 12,* la precisión alcanza su valor máximo. Corrimos un código nuevamente para calcular los valores de cada medida de performance, y conseguimos estos resultados:

- *Accuracy*: 0.9489285714285715
- *Precision*: 0.9313244569025928
- *Recall*: 0.9672489082969432
- *F1 score*: 0.9489468047126026

De hecho, cada medida aumentó, por lo que estos píxeles son mejores en todo sentido. No aumentó la precisión en detrimento de alguna otra medida. Entonces, nos preguntamos: al unir ambos conjuntos de píxeles, ¿veremos una mejora notable? Nuevamente, a través de un gráfico, respondimos nuestra duda:

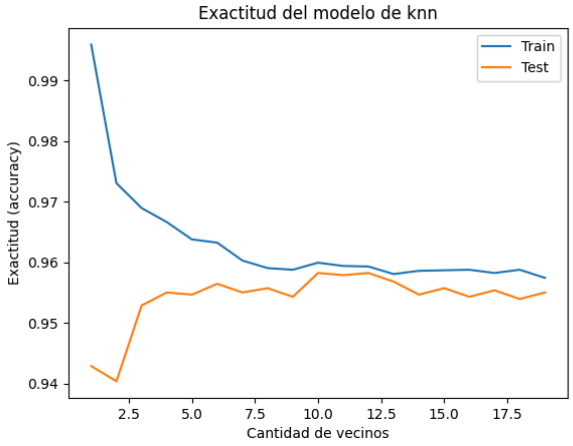


Gráfico 4: precisión del modelo KNN en función de k con los píxeles 40, 68, 96, 70, 126, 526 y 554

La respuesta es sí, la precisión aumentó, pero **no sustancialmente.** Analicemos esto con los puntajes de performance cuando *k = 12*, que es el pico de la precisión.

- *Accuracy*: 0.95821428571428
- *Precision*: 0.9596317280453258
- *Recall*: 0.9575971731448764
- *F1 Score*: 0.958613371064733

Podemos notar que el *Recall* cayó un poco, mientras las demás aumentaron. Es adecuado argumentar que este conjunto de píxeles es mejor, pero que valga la pena es cuestionable. El código tarda más en ejecutarse al analizar más píxeles, y la información que algunos de ellos brindan es redundante. Agregar más píxeles al subconjunto de atributos no traería buenos resultados tampoco, por esta misma razón. Hemos concluido, entonces, que el set más óptimo que hemos encontrado de píxeles es aquel mencionado durante el *análisis exploratorio*.

Predicciones del modelo

En este apartado, relataremos cómo entrenamos a nuestro modelo de clasificación, y qué performance tuvo luego de testearlo. En primer lugar, separamos un conjunto de entrenamiento y de validación. Utilizando el primer conjunto, ajustamos un modelo de decisión, registrando su *accuracy* y otras medidas de performance, variando la altura (*k)*. Como veremos más adelante, obtuvimos mayor exactitud a mayores valores de *k*. Por ejemplo, si *k = 10,* conseguimos una accuracy de 0.78, mientras que si *k* = 6, la exactitud bajaba a 0,69.

Por otro lado, realizamos un entrenamiento de validación cruzada. Separamos el conjunto de desarrollo en 5 *folds* -nuevamente, variando la altura-, y probando distintos criterios de clasificación, como *gini* y *entropía.* De esta forma, realizamos 5 testeos en cada altura, porque el *fold* de testeo iba rotando, y calculamos un promedio de la performance por cada altura. Estos fueron los resultados que obtuvimos:

Altura utilizada	Accuracy con criterio entropía	Accuracy con criterio gini
1	0,1987	0,1988
2	0,3410	0,3309
3	0,4990	0,4766
4	0,6560	0,6183
5	0,6859	0,6764
6	0,6968	0,6948
7	0,7217	0,7141
8	0,7395	0,7285
9	0,7476	0,7446
10	0,7503	0,7474

Aunque muy leve, hay una mejora utilizando el criterio de la *entropía* en vez de *gini gain,* por lo que nos decantamos por él a la hora de evaluar la performance de nuestro modelo. El último paso fue, entonces, entrenar a nuestro modelo en todo el conjunto de desarrollo, y evaluar su *accuracy* sobre el conjunto *held-out,* que hasta este punto no lo habíamos utilizado en lo absoluto. A continuación, veremos la matriz de confusión que obtuvimos, junto a distintas medidas de exactitud:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	993	6	46	156	17	4	150	3	16	3
1	4	1301	6	64	9	5	9	0	4	0
2	10	1	998	32	240	1	113	0	22	0
3	59	45	24	1161	99	1	45	1	9	5
4	18	2	158	108	870	0	185	0	15	1
5	5	5	0	8	1	1000	1	327	27	75
6	194	1	208	122	257	1	570	2	52	0
7	2	0	1	0	0	94	0	1159	10	93
8	15	2	37	11	15	11	22	16	1206	7
9	6	0	2	6	1	50	0	86	7	1276

Matriz de confusión: en las filas, las clases verdaderas. En las columnas, las clases predichas.

Con esta matriz de confusión a nuestra disposición, podemos evaluar distintas medidas de performance sobre el modelo que diseñamos. En primer lugar, destacamos que obtuvimos una accuracy de aproximadamente 75%, un porcentaje relativamente alto, si consideramos que teníamos 14.000 prendas a categorizar en el conjunto *held-out*. Por otro lado, podemos ver que hay muchos valores relativamente bajos, incluso algunos ceros, ¡lo que evidencia que nuestra selección de píxeles para separar aquellas clases era adecuada!

También verificamos una afirmación que habíamos postulado anteriormente: la proporción de la cantidad de muestras de cada clase sobre el total está bien balanceada, pues la suma de cada fila de la matriz da resultados aproximados.

Pero nuestro modelo no es infalible, por lo que podemos ver especialmente en las celdas marcadas en rojo. El modelo clasificó muchas muestras de la clase *5* como clase *7,* y tampoco supo distinguir muy bien entre las clases *2, 4 y 6*. Consideramos que esto se debe a tanto nuestra selección de píxeles, como también a las similitudes entre las clases, lo que complejizó su distinción.

Por otro lado, también analizamos el resultado más alto y más bajo por cada una de las medidas de performance -utilizadas anteriormente sobre el modelo *Knn*- para interpretar mejor en qué aspecto de la clasificación el modelo es más falible.

- *Recall*:
  - Más alto: Clase 1 = 0,92
  - Más bajo: Clase 6 = 0,4
- *Precisión*:
  - Más alto: Clase 1 = 0,96
  - Más bajo: Clase 6 = 0,52
- *F1 score*:
  - Más alto: Clase 1 = 0,94
  - Más bajo: Clase 6 = 0,455

Es evidente que el modelo tiene una exactitud sobresaliente a la hora de distinguir prendas de la clase 1 de las demás, pero falla considerablemente en todo sentido con la clase 6: detecta menos de la mitad de las prendas, y además califica erróneamente a otras. Quizás, con un par de píxeles más que separen a esta clase, la exactitud mejoraría considerablemente.

Conclusiones

Hemos logrado, entonces, luego de una profunda exploración del dataset original, armar dos modelos de clasificación que realicen predicciones con una exactitud razonable.

Como habíamos anticipado antes de armar el modelo, las clases cuyas imágenes eran muy similares entre sí fueron las más difíciles de clasificar, y las causantes de la mayoría de errores del árbol de clasificación. Por otro lado, las prendas con una forma más peculiar, como los pantalones, fueron las más fáciles de separar de las demás, y las que el modelo clasificó con más exactitud, precisión y *recall.*

También notamos que nuestro modelo de clasificación *Knn* registró una exactitud exorbitante en comparación con el árbol. Por un lado, porque el primero únicamente debió separar prendas de 2 clases en vez de 10, pero también, porque las formas de dichas clases eran bastante diferentes.

Aunque los modelos no son perfectos, con unos pequeños ajustes en los parámetros (o hiper parámetros), ya sea aumentando la cantidad de atributos o realizando una selección más adecuada de ellos, es muy probable que su exactitud alcance valores más altos, e incluso puedan llegar a utilizarse en la práctica.