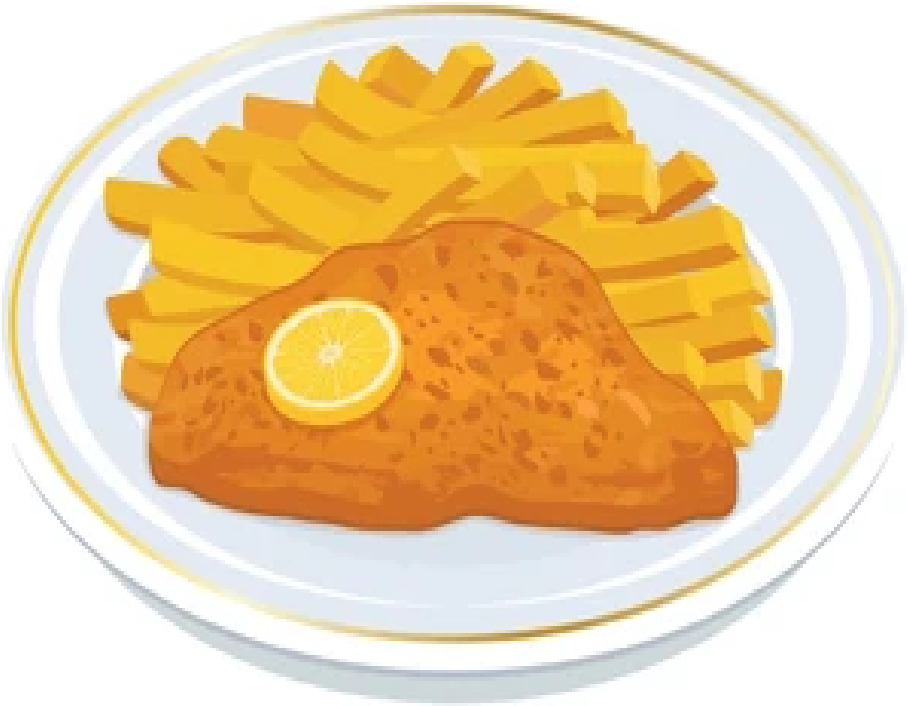
Laboratorio de Datos
Trabajo Práctico 02

2do. Cuatrimestre - 2025

GRUPO:
“IMPORT_MILANESAS”



Integrantes: Dulio Joaquin, Risuleo Franco, Perez Sotelo Martina

Introducción:

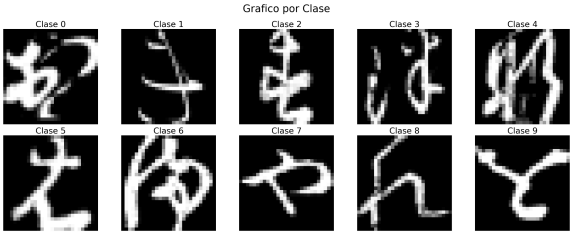
Se nos propuso trabajar con un conjunto de datos de imagen “Kuzushiji-MNIST”. Este dataset está basado en uno muy famoso “MNIST”. Queremos explorar el conjunto de datos e implementar una clasificación binaria y multiclase.

Análisis exploratorio de los datos:

Este dataset cuenta con 70.000 imágenes de tamaño 28x28 y viene acompañado por un dataset de mapeo, donde a cada letra japonesa le corresponde una etiqueta numérica que va del 1 al 10.

Este conjunto de datos viene dado en forma de tabla csv, donde podemos ver que:

- Hay 784 atributos que indican al **valor** de intensidad que toman los píxeles de la imagen, este valor está en un rango de entre 0 y 255. Son tipos de datos continuos.
- El último atributo nos indica la **etiqueta numérica** de la letra japonesa representada en esa fila. Es un dato categórico.
- Hay 70.000 filas, así cada fila es una imagen distinta. Se debe hacer una **reestructuración de los datos** ordenándolos en 28 filas y 28 columnas para poder visualizar claramente la imagen.

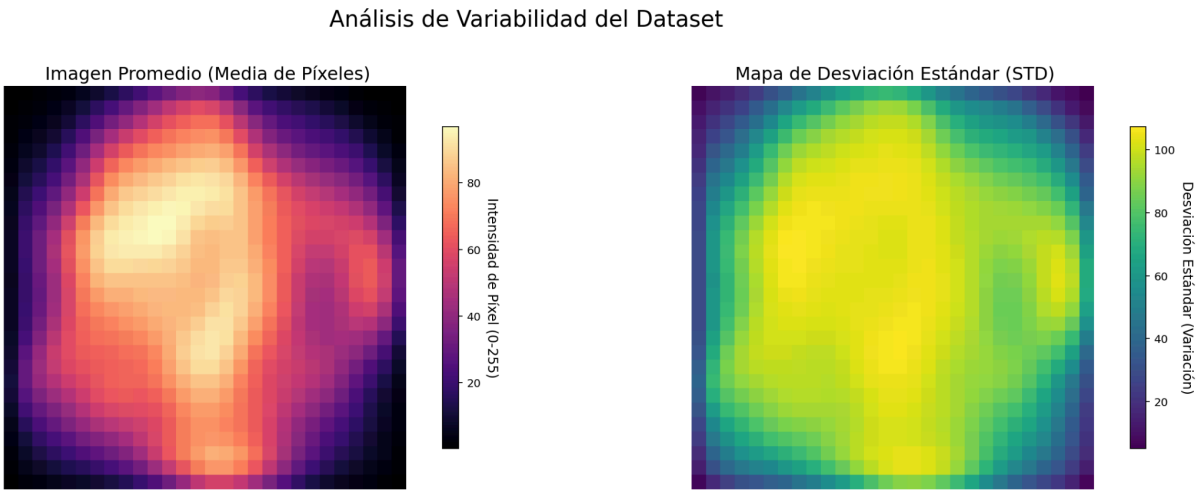


- El dataset está **balanceado**, presenta 7000 imágenes de cada clase. Esto es ideal para el trabajo con machine learning.

Haciendo un análisis más exploratorio por medio de visualizaciones se puede ver:

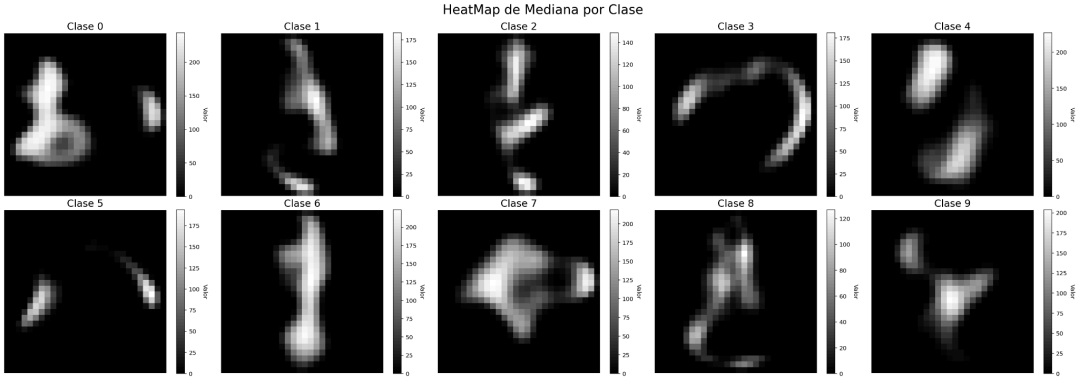
- **Alta variabilidad presente en el dataset:** La variabilidad del dataset es alta. Solamente los píxeles de las esquinas suelen ser consistentes, al ser píxeles que permanecen apagados. El resto de ellos presenta una desviación estándar de entre 40 y 100. Se puede ver en el gráfico que toda la parte central de la imagen presenta grandes variaciones (color amarillo) y en el contorno de la imagen promedio sigue habiendo una

variacion notable (color verde).



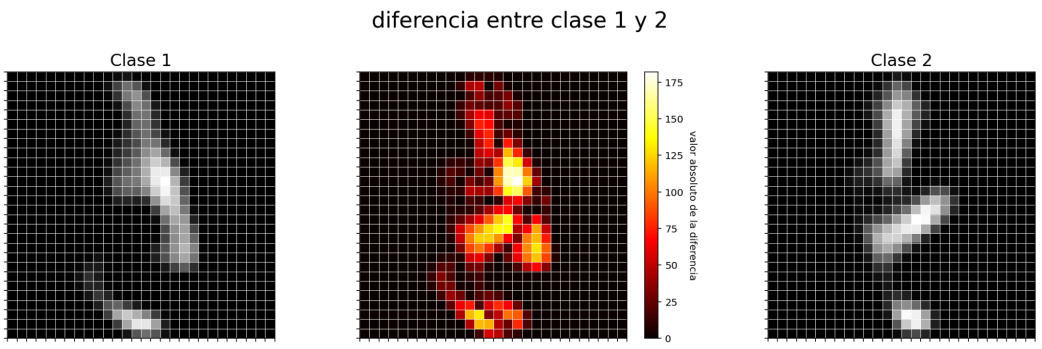
Esto indica que tenemos que trabajar con todos los atributos del dataset, ya que presentan una alta variabilidad. También podríamos descartar los píxeles de las esquinas por su poca variabilidad, pero no impacta demasiado en el tamaño del dataset. Elegimos mantener el dataset tal cual está dado.

- **Diferencias y coincidencias entre imágenes:** A partir de calcular la mediana de los valores de píxel de cada clase podemos comparar en cuanto difieren dos imágenes de clases distintas.



Si tomamos la comparación entre las clases 1 y 2, y la comparación entre las clases 2 y 6, podemos ver en las zonas de color que los gráficos de las medianas difieren un poco más entre las clases 2 y 6 que entre las clases 1 y 2

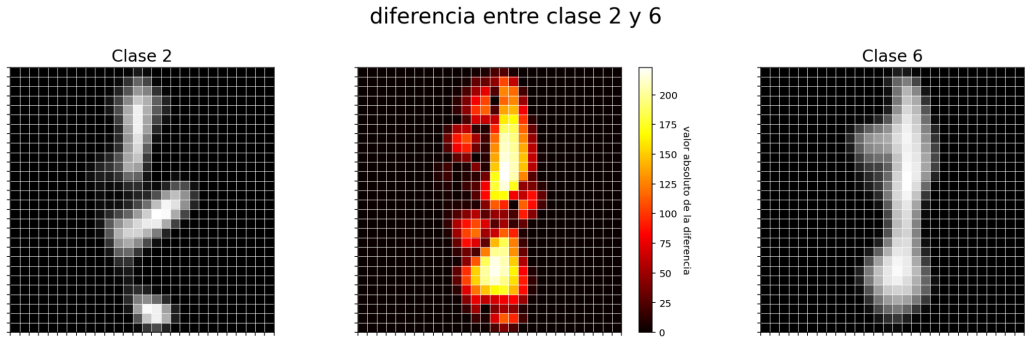
ya que en el segundo gráfico podemos ver que las zonas más amarillas son aquellas que presentan mayor diferencia. y comparándola con el gráfico de la derecha el gráfico 2 presenta más zonas de grandes diferencias. (puedo usar std)



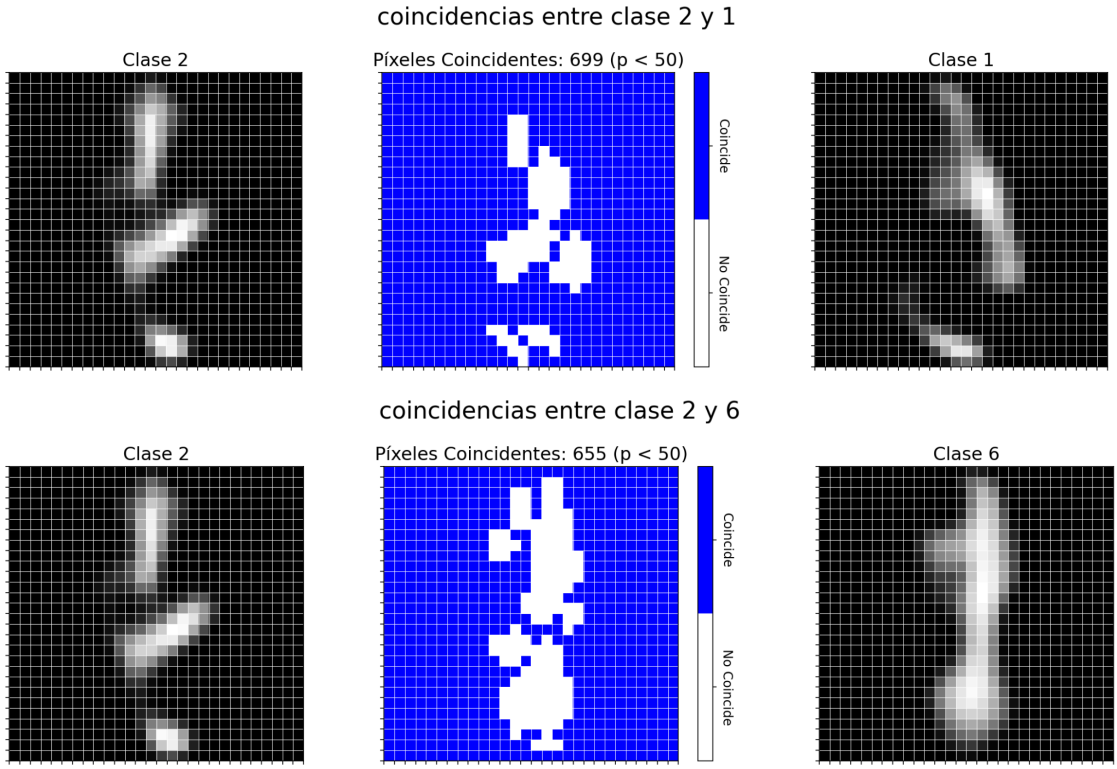
Si queremos orientar nuestro análisis más a hacia buscar coincidencias o ver qué tan parecidos son los gráficos de las medias, podemos considerar la siguiente ecuación. Sea A el gráfico de la izquierda y B el gráfico de la derecha, i el número de pixel y p un parámetro,

si $|pixel_{Ai} - pixel_{Bi}|<p$, entonces considero como parecidos/coincidentes a $pixel_{Ai}$ y $pixel_{Bi}$.

Así uno podría considerar que los valores iguales entre píxeles son aquellos tales

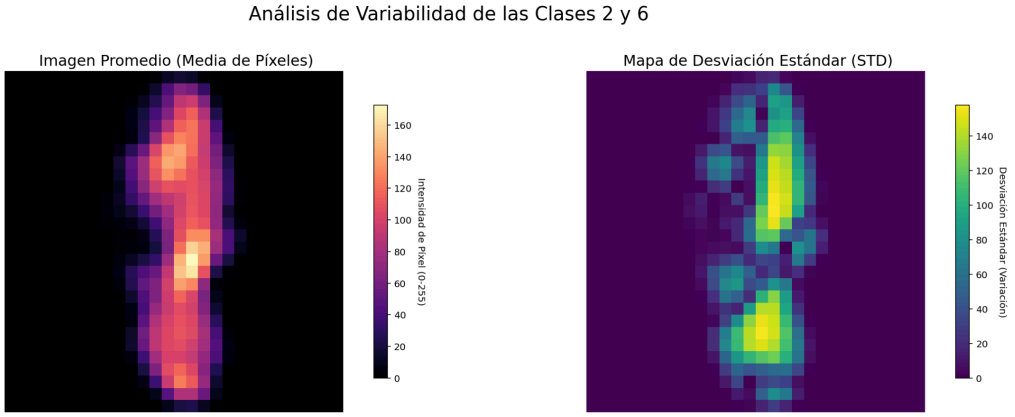


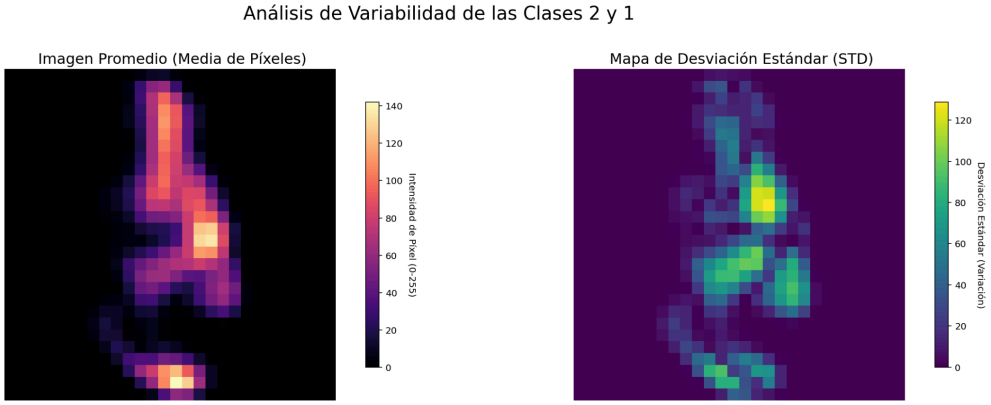
que p=1 , en el siguiente gráfico probamos con un p=50.



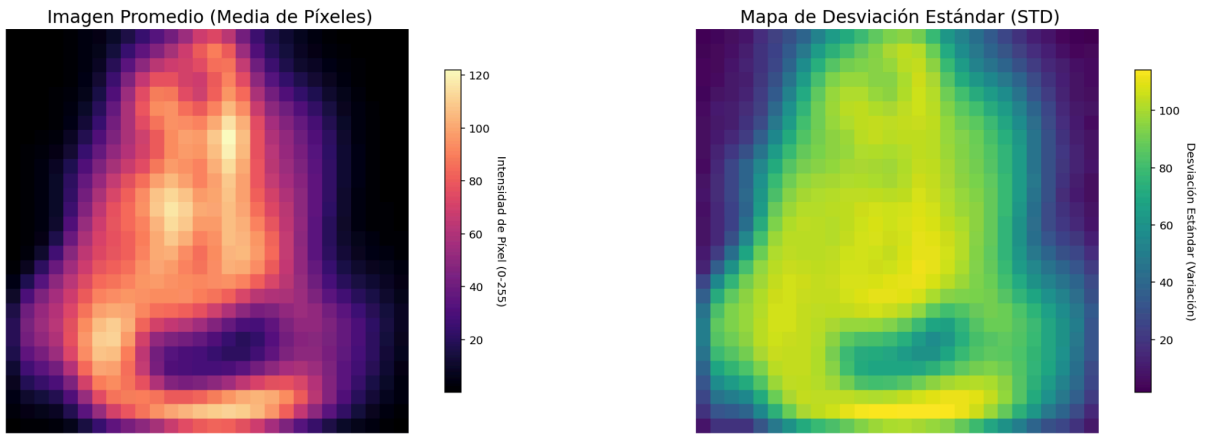
Las imágenes entre las clases 2 y 1 son más parecidas que las imágenes entre las clases 2 y 6. Esto, a la hora de entrenar nuestro modelo, puede suponer una complicación, ya que si dos imágenes tienen valores parecidos de píxeles en una zona, habría que tener el cuidado de que el modelo no se guíe solamente por esta, sino que tome zonas más amplias.

También podemos considerar la variabilidad de cada mediana, vemos que los gráficos siguientes coinciden y apoyan el análisis que hicimos con los gráficos anteriores.



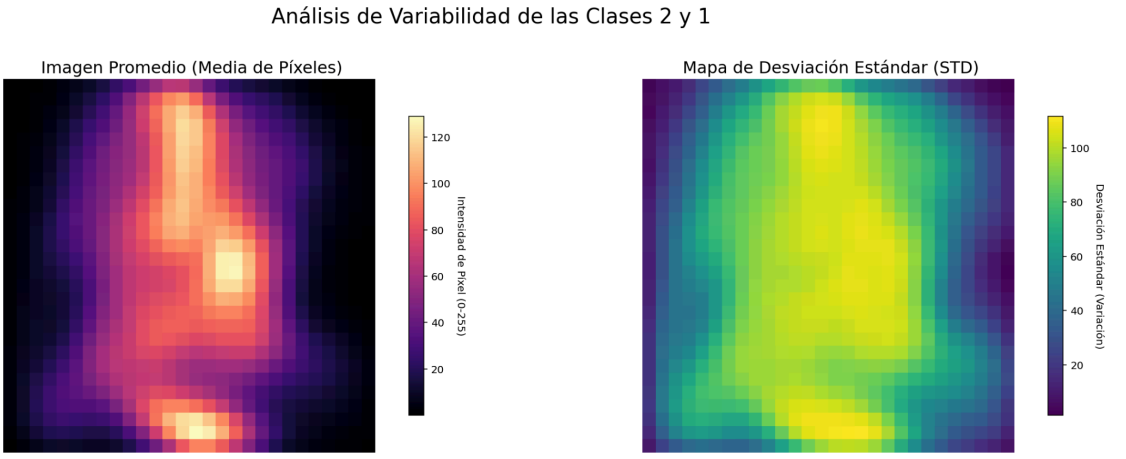


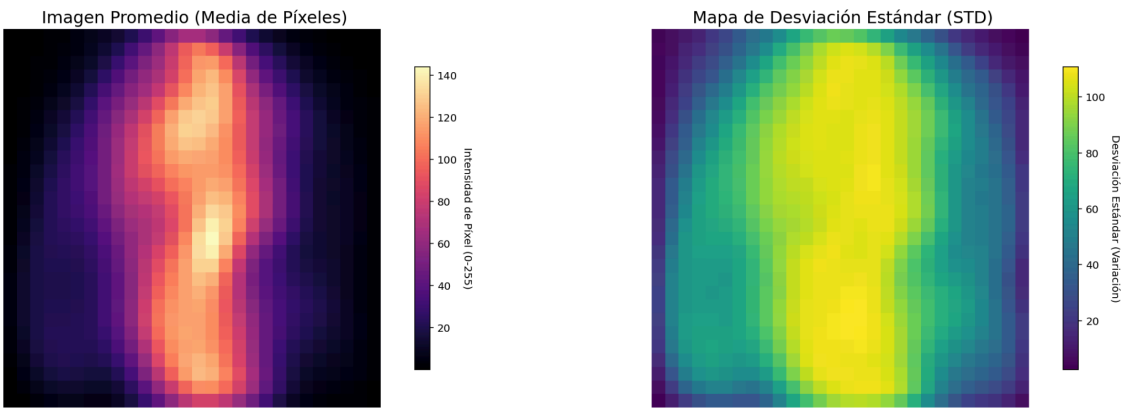
- **Variabilidad por clase:** También podemos analizar la variabilidad de cada píxel dentro del dataset de una sola clase.



así vemos que, dentro de una misma clase la variabilidad sigue siendo alta. En la clase 8, por ejemplo, vemos pocas zonas donde la variabilidad es baja. Esto significa que dentro del dataset hay gran variedad de imágenes distintas correspondientes a la clase 8. Esto va a suponer un desafío para nuestro modelo ya que los valores de píxel no suelen ser constantes entre clases.

- **Variabilidad entre dos clases:** Siguiendo un poco la línea de querer encontrar que tan coincidentes son dos clases distintas, podemos analizar qué pasa con la variabilidad entre dos clases.





Analizando la clase 2 contra las clases 1 y 6, seguimos viendo una alta variabilidad en ambos casos, pero es interesante ver como cambia las zonas de mayor interés para el análisis entre las dos clases.

Clasificación Binaria:

Para este experimento, tomamos las clases 4 y 5. Nuestro objetivo es entrenar y obtener el mejor modelo KNN posible para clasificar según las dos clases. Para esto, separamos los datos de entrenamiento y los datos de testing (80/20), fijamos un K=5 y probamos con distintos conjuntos de píxeles para ver como varían los resultados. Primero, exploramos los resultados con atributos de distintas características:

- Del centro (lejos de los bordes de las imágenes, donde se espera una variabilidad de datos mayor).
- De las esquinas (donde uno no suele escribir).
- Seguidos.
- Aleatorios.

Analizamos el valor de accuracy dado. Luego, analizando un poco la variabilidad de los datasets, probamos cuáles eran los resultados al elegir atributos con mayor nivel de variabilidad entre ambas clases, nos quedamos con los atributos cuya desviación estándar es mayor a 111. Vemos que así mejora el accuracy. Tras una exploración más profunda, decidimos utilizar el discriminante lineal de Fisher. Esta métrica nos ayuda a determinar qué tan bien un atributo individual puede discriminar dos clases. Nos quedamos con aquellos atributos que arrojan un resultado mayor a 1.

En la siguiente tabla se ven, en orden de cómo fueron listados anteriormente (centro, esquinas, seguidos, aleatorios, variabilidad alta, y variabilidad alta entre clases pero baja en una misma clase) los conjuntos de atributos utilizados, la cantidad que probamos y el accuracy obtenido.

Cantidad Atributos	Atributos Usados	Accuracy
4	[0, 27, 28, 55]	0.5128571429
4	[56, 83, 84, 168]	0.5135714286
4	[17, 62, 92, 120]	0.7660714286
4	[444, 445, 446, 447]	0.7067857143
4	[578, 612, 499, 702]	0.6835714286
4	[416, 370, 360, 388]	0.7146428571
4	[94, 122, 149, 150]	0.8328571429
5	[0, 27, 28, 55, 56]	0.515
5	[56, 83, 84, 168, 783]	0.5153571429
5	[17, 62, 92, 120, 355]	0.7767857143
5	[444, 445, 446, 447, 448]	0.71
5	[578, 612, 499, 702, 550]	0.5882142857
5	[416, 151, 370, 360, 388]	0.8171428571
5	[177, 178, 205, 206, 122]	0.8478571429
7	[0, 27, 28, 55, 56, 83, 84]	0.5132142857
7	[56, 83, 84, 168, 196, 224, 252]	0.5585714286
7	[17, 62, 92, 120, 264, 148, 404]	0.7939285714
7	[444, 445, 446, 447, 448, 449, 450]	0.7085714286
7	[578, 612, 499, 702, 123, 389, 12]	0.7835714286
7	[151, 388, 398, 416, 370, 360, 389]	0.8117857143
7	[149, 178, 206, 150, 94, 205, 122]	0.8642857143

Remarcado en amarillo podemos ver nuestro mejor modelo basándonos en su resultado de accuracy. Se puede ver como la selección de los atributos basados en la variabilidad tiene una mejor performance.

Después de analizar los atributos, queremos encontrar la mejor cantidad de atributos para nuestro modelo y el mejor K (cantidad de vecinos). Creamos una función que vaya iterando y se quede con el mejor K para cada cantidad de atributos. Tomamos una cantidad de atributos de un rango entre 2 y 30 con step 5. En la siguiente tabla podemos ver los resultados ordenados de mayor a menor por accuracy.

Cantidad Atributos	Atributos Usados	Mejor K	Accuracy
20	[23, 375, 712, 411, 325, 294, 660, 414, 606, 490, 626, 91, 462, 579, 385, 70, 760, 704, 463, 564]	12	0.9253571429
20	[418, 233, 436, 623, 334, 147, 281, 315, 209, 146, 232, 598, 94, 357, 668, 222, 234, 351, 619, 415]	12	0.9253571429
20	[416, 13, 496, 603, 557, 220, 155, 37, 713, 30, 49, 168, 748, 523, 9, 94, 500, 178, 60, 590]	12	0.9125
20	[403, 107, 29, 571, 6, 660, 602, 450, 341, 64, 174, 711, 632, 413, 442, 196, 468, 618, 356, 431]	12	0.9078571429
20	[326, 29, 515, 409, 20, 518, 276, 441, 341, 197, 528, 199, 280, 204, 357, 364, 617, 742, 314, 85]	12	0.9078571429
20	[340, 126, 531, 307, 615, 773, 456, 318, 150, 82, 120, 230, 27, 41, 510, 417, 311, 227, 63, 602]	27	0.9039285714
20	[566, 102, 93, 705, 744, 98, 8, 268, 778, 347, 292, 692, 180, 190, 407, 318, 269, 532, 474, 267]	7	0.9035714286
15	[350, 118, 684, 40, 205, 669, 173, 648, 315, 419, 365, 433, 423, 653, 2]	17	0.9014285714
20	[485, 203, 14, 641, 25, 275, 371, 286, 527, 717, 412, 637, 45, 20, 96, 30, 504, 144, 256, 524]	12	0.9010714286
15	[154, 763, 748, 77, 271, 597, 518, 679, 530, 130, 106, 693, 438, 540, 658]	12	0.8985714286
20	[435, 779, 328, 578, 246, 104, 534, 23, 735, 8, 502, 516, 141, 715, 439, 58, 532, 524, 241, 559]	12	0.8907142857
15	[112, 621, 778, 309, 288, 315, 748, 389, 489, 774, 423, 231, 761, 467, 505]	7	0.8875

La tabla completa se encuentra en carpeta Resultados_Binario

Clasificación Multiclase:

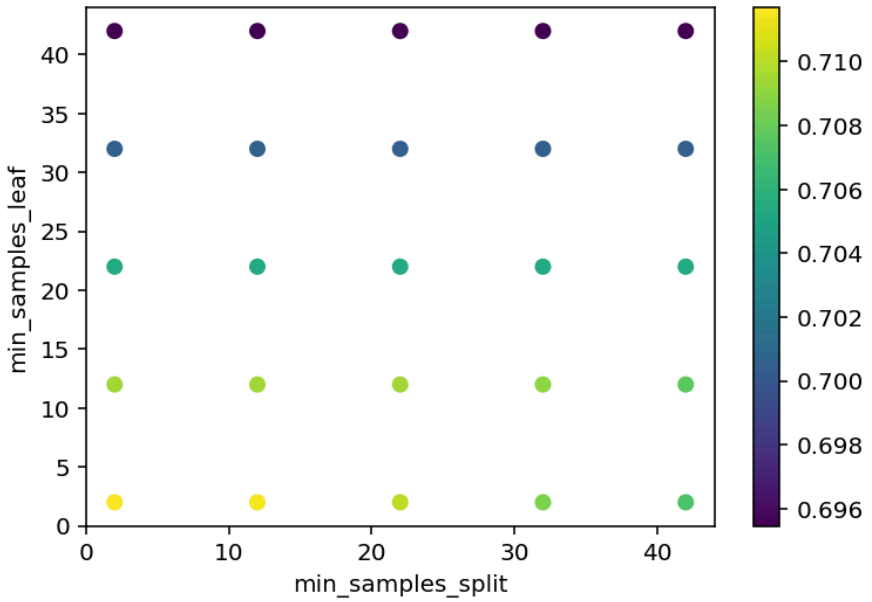
En busca de un modelo que logre predecir entre las diez clases que se presentan en el dataset dado, entrenamos un árbol de decisión.

Como primer paso, dividimos los datos en dos partes: los datos de entrenamiento (80%) y los datos de testing (20%). A continuación, probamos entrenar distintos modelos, variando la profundidad entre 1 y 10. Dividimos el conjunto de entrenamiento en dos subconjuntos, nuevamente de 80% y 20%. Esto se hace con el objetivo de evitar usar los datos de testing a la hora de elegir un modelo, ya que implicaría el uso indirecto de estos para entrenar. Así, del conjunto de entrenamiento tenemos un nuevo subconjunto de testeo, más pequeño.

Como resultado, al restringirnos a profundidades no mayores a 10, vimos que el modelo con mejor exactitud fue el de 10, con un 70%. Sin embargo, no encontramos el punto a partir del cual una mayor profundidad comienza a afectar negativamente las

predicciones del modelo, por lo que decidimos continuar entrenando modelos con valores mayores. Encontramos que el pico de desempeño del modelo se encuentra entre 30 y 40, habiendo niveles de precisión cercanos al 73%.

Luego de hallar un nivel de profundidad máxima con los demás hiperparámetros constantes, buscamos entrenar más modelos a modo de prueba, esta vez variando todos los hiperparámetros de interés: profundidad máxima (entre 1 y 10), criterio (*Gini* o *Entropy*), max_samples_split y max_samples_leaf. Como era de esperarse, aquellos modelos con profundidad 10 fueron significativamente mejores que los demás. En general, se aprecia que Entropy tiende a dar resultados levemente mejores que Gini, pero los valores de max_samples_split y max_samples_leaf no parecían afectar mucho. Con el objetivo de analizar mejor el impacto de estos, entrenamos más modelos con una profundidad máxima de 10 y utilizando el criterio de Entropy. Así, generamos el siguiente gráfico.



Vemos que, en líneas generales y para nuestro dataset con dicha profundidad y criterio, menores valores de ambos parámetros resultan en desempeños levemente mejores.

Tras todo este análisis, encontramos que el mejor modelo posible sería uno con una profundidad de 10, Entropy como criterio, y valores mínimos para max_samples_split y max_samples_leaf (2 y 1, respectivamente). Finalmente, utilizando el conjunto de datos de testing que apartamos al principio de este proceso, hallamos que se obtiene una precisión del 71,2%.

Matriz de Confusión: Clasificación Multiclase

Podemos observar como nuestro modelo arroja una mejor predicción para algunas clases como las clases 0 y 3.

	Pred 0	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5	Pred 6	Pred 7	Pred 8	Pred 9
Real 0	1150	10	9	30	42	34	21	43	38	20
Real 1	22	892	95	52	62	20	69	15	128	44
Real 2	11	86	816	83	59	27	192	10	76	45
Real 3	25	34	28	1177	32	50	9	31	30	20
Real 4	63	69	44	38	887	27	79	21	36	56
Real 5	37	41	40	76	47	1133	40	23	41	11
Real 6	17	94	71	25	73	36	936	24	51	20
Real 7	45	18	32	39	54	11	37	1126	23	33
Real 8	34	96	66	48	15	19	73	10	965	25
Real 9	41	104	97	46	81	8	33	47	50	931

Reporte completo por clase:

clase	precisión	recall	f1-score	support
0	0.8	0.82	0.81	1397
1	0.62	0.64	0.63	1399
2	0.63	0.58	0.6	1405
3	0.73	0.82	0.77	1436
4	0.66	0.67	0.66	1320
5	0.83	0.76	0.79	1489
6	0.63	0.69	0.66	1347
7	0.83	0.79	0.81	1418
8	0.67	0.71	0.69	1351
9	0.77	0.65	0.7	1438

Conclusiones:

Luego de haber analizado el dataset y desarrollar los experimentos, concluimos que el mejor modelo multiclase es aquel que tiene una profundidad máxima de 10, utiliza *entropy* como criterio, y tiene valores mínimos para min_samples_leaf y min_samples_split. Por otro lado, el mejor modelo binario es aquel que se basa en los píxeles cuya variabilidad entre clases sea alta, pero dentro de cada clase sea baja. Esto evita tener en cuenta atributos que no suelen aportar mucha información. De esta manera, el costo computacional es más bajo y el rendimiento no se ve afectado por el ruido que pudieran generar estos atributos irrelevantes.

Bibliografía:

- Clanuwat, Tarin, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. "Deep learning for classical japanese literature." arXiv preprint arXiv:1812.01718 (2018).