DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

Clasificación y Validación Cruzada

10 de Marzo de 2024 Laboratorio de Datos

Integrante	LU	Correo electrónico
Teplizky, Gonzalo	201/20	gonza.tepl@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Resumen

El presente trabajo tiene por objetivo poder predecir, dada una imágen particular, qué letra representa en el lenguaje de señas *MNIST*. Para ello se utilizó el dataset *Sign Language MNIST* [1].
A partir de este fue realizado análisis exploratorio inicial, y luego se desarrollaron distintos modelos predictivos de clasificación que intentan responder, dada una imágen correspondiente a una letra en *MNIST*, de cuál se trata.

Palabras clave: *Sign Language MNIST*, *Imágen*, *Letra*, *Exploración*, *Visualización*, *Análisis*, *Clasificación*, *Patrones*, *Modelo*, *Predicción*

Índice

[1] https://www.kaggle.com/datasets/datamunge/sign-language-mnist

1. Introducción - Presentación del problema

En una sociedad moderna y globalizada como la nuestra, la inclusión de todos los ciudadanos se encuentra a la orden del día. En este contexto, los lenguajes de señas constituyen una pieza fundamental en la búsqueda de la igualdad de condiciones para todas las personas, ya que fomentan que quienes tienen cierta discapacidad auditiva puedan comunicarse e integrarse en plenas condiciones a las actividades del día a día. Además, su aprendizaje por parte de personas sin discapacidad fomenta la comprensión y la diversidad, creando así un entorno más inclusivo y colaborativo. En este contexto, resulta de vital importancia que la tecnología esté adaptada a estas potenciales necesidades de sus usuarios. En particular, se busca poder enseñarles a interpretar todo tipo de lenguaje que pueda ser utilizado en una sociedad determinada.

Tomando esta finalidad, procedemos a evaluar y desarrollar ciertos modelos que puedan servir eventualmente como guía para que una computadora pueda aprender a reconocer e interpretar las distintas letras que posee el lenguaje de señas *MNIST*. A partir del dataset provisto pudimos obtener un conjunto de imágenes que contiene distintas representaciones de las letras en *MNIST*. Cada imágen corresponde a una letra única, expresada en $28x28$ píxels en escala de grises. Por cada letra contamos con un conjunto considerable de imágenes que las representan.

Esto plantea la primera diferencia sustancial con los demás datasets que venimos analizando, donde los datos representa texto, palabras o conjuntos de letras. Es por esto que el enfoque para analizarlos debe ser distinto. Cada imágen de este dataset se representa mediante 785 atributos distintos. El primer atributo es el *label*, es decir, la clase a la cual cada letra pertenece. Cada clase se encuentra expresada como un número entre el 0 y el 25, donde cada uno corresponde a una letra entre la A y la Z. En particular el dataset no contiene imágenes de las letras J y Z, es decir labels 9 y 25, dado que estas letras expresadas en *MNIST* presentan movimiento. Cada uno de los 784 atributos restantes corresponde a un pixel determinado de la imágen, expresado como un número entre 0 y 255 que determina la tonalidad en escala de grises que ese color tiene.

Partiendo de este conocimiento, la exploración de estos datos debe ser radicalmente distinta a lo que estamos acostumbrados, es decir, debe ser adecuada a los parámetros de cada imágen con los que estamos trabajando.

A simple vista se puede apreciar que cada seña se encuentra localizada al centro de cada imágen, por lo que en un principio podríamos suponer que es posible quedarnos con un fragmento de cada imágen para visualizar de que letra se trata la representación. Pero mirando detenidamente otras imágenes de la misma clase de letra, notamos que en muchas de estas las señas se encuentran más corridas sobre el eje X, tanto en una dirección como en la otra. Y algo similar ocurre al observar la distribución de los píxels sobre el eje Y. Para estar seguros, consideremos la varianza que estas imágenes tienen en los bordes.

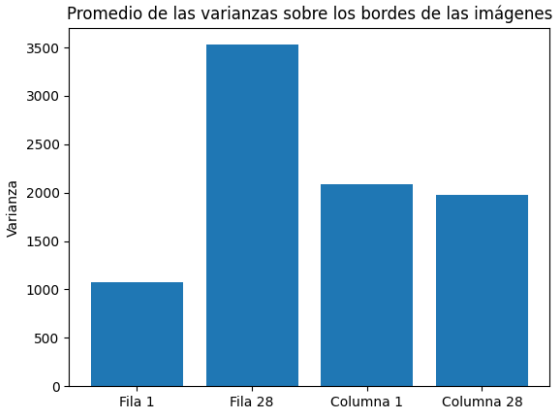

Figura 1

Para hacer este gráfico, consideramos 100 representantes de cada letra y calculamos la varianza para cada píxel $Var[(i,j)]$ sobre la totalidad de las muestras. Después, consideramos el promedio de estos valores en los bordes; es decir, calculamos $Prom_{\text{Fila 1}} = \dfrac{\sum_{j=1}^{28} Var[(1,j)]}{28}$, y de la misma manera hacemos el promedio de la última fila, y de la primera y última columna. La idea es que, a valores bajos, se podría pensar que dado que estamos trabajando con imágenes de todas las clases, uno podría descartar la fila/columna asociada a dicho valor. Sin embargo, ya sea porque es difícil interpretar la magnitud de estos valores o porque efectivamente es necesario preservar las imágenes en su totalidad, no pudimos concluir que haya partes que se puedan descartar. Además, guíandonos por nuestras propias observaciones, creemos lo contrario: todos las partes de una imagen son fundamentales para su clasificación por medio de cualquier modelo predictivo.

Podemos notar que en este lenguaje hay letras que se expresan de forma muy similar. La A, la E y la M tienen diferencias muy sutiles que responden a donde se coloca el dedo pulgar y/o el dedo meñique. Entre la G y la H la única diferencia clara es que una señala con el índice y la otra le agrega el dedo del medio. Este patrón se repite con la V y la W, dado que una toma el anular y la otra no. Y finalmente entre la O y la F la sutileza se encuentra en cuanto se juntan los dedos índice con pulgar, formando un círculo o una forma más ovalada.
Para ejemplificar esto observemos las clases E, L y M:

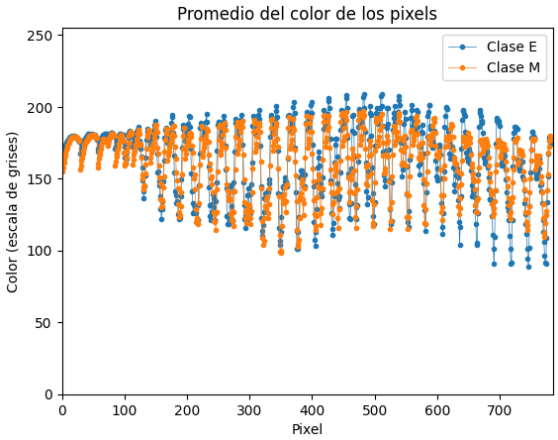

Figura 2

En este primer gráfico decidimos mostrar el valor promedio del color por píxel de las imágenes correspondientes a las letras E y M. Podemos observar un par de cuestiones: vemos que pareciera haber una alta coincidencia entre los valores en y correspondientes a los píxeles 0 a 150 de ambas clases. Además, si bien la diferencia en y entre las letras aumenta a mayor valores en x, a priori se podría pensar que las señas son parecidas entre sí. Habiendo dicho esto, hay que tener un poco de cuidado con esta conjetura: al usar esta métrica y sacar conclusiones a partir de sus resultados, estamos suponiendo que en este caso el promedio es una medida de tendencia que preserva con cierta fidelidad la totalidad de los valores que está caracterizando. Notemos que no estamos teniendo en cuenta la varianza por píxel de las imágenes de una misma clase, dato que podría cambiar rotundamente el análisis (o no) si fuera considerado. Sin embargo, nos inclinamos a pensar que la tonalidad por píxel no varía demasiado para estas letras, habiendo comparado las señas a ojo y viendo que se está representando lo mismo en cada imagen (en este caso, una misma letra).

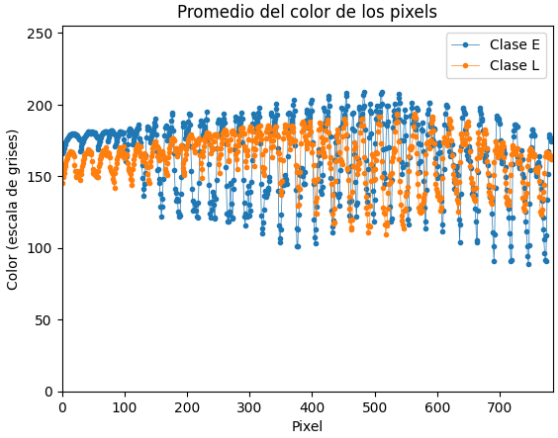

Figura 3

Observemos que en el segundo gráfico los valores para un mismo x parecieran ser más distantes que en el primero, cosa que nos induce a pensar que la seña de la letra L es más parecida a la seña de la M que de la E. Nuevamente hacemos hincapié en el hecho de que estamos trabajando con una medida de tendencia, no la totalidad de los valores, y por lo tanto hay información que se pierde y puede llevar a conclusiones erróneas.

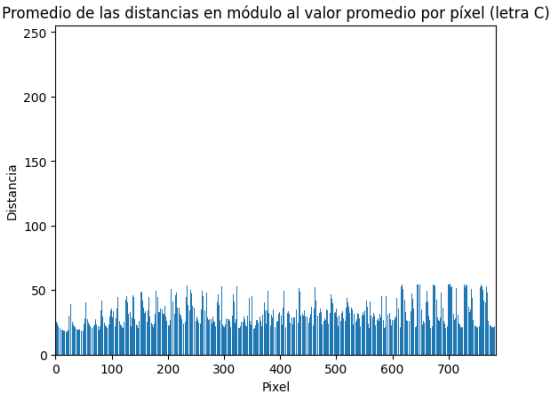

Figura 4

Para el ítem (c) decidimos tomar el promedio de los valores de las imágenes correspondientes a la letra C para cada píxel (llamemos a esta tupla de promedios P, con P de tamaño 784), y hacer el promedio de las distancias en valor absoluto entre las imágenes y la tupla P. Para que se entienda mejor, si Im{C} es el conjunto de imágenes con este label, y $X_k^{(i)}$ es el k-ésimo píxel de una imágen arbitraria $X^{(i)} \in \text{Im}\{\text{C}\}$, calculamos y graficamos $D := (\overline{|X_1^{(i)} - P_1|}, \overline{|X_2^{(i)} - P_2|}, \ldots), D \in R^{784}$ con respecto a $k : 1 \leq k \leq 784$, donde $\overline{|X_k^{(i)} - P_k|} = \frac{\sum_{X^{(i)} \in \text{Im}\{\text{C}\}} |X_k^{(i)} - P_k|}{|\text{Im}\{\text{C}\}|}$. También se puede pensar como una especie de varianza por píxel, haciendo la diferencia en módulo en vez de al cuadrado.
Elegimos esta métrica porque nos pareció conveniente y más fácil de interpretar que la varianza a la hora de evaluar cuán parecidas son las imágenes. Yendo al análisis del gráfico, notamos que las distancias suelen ser de 30 a 50 píxeles al promedio; podríamos pensar que, dada la cantidad considerable de samples con la que estamos trabajando y el rango de valores que vemos en el gráfico, la diferencia entre 2 imágenes elegidas arbitrariamente con este label suele no ser despreciable.

2. Desarrollo, Modelado y Observaciones

En esta sección hablaremos sobre el desarrollo de los ejercicios (2) y (3). En particular, nos focalizaremos en explicar el porqué de las decisiones que fuimos tomando para la construcción de los modelos, y profundizar en el análisis mediante gráficos que nos permitirán entender e interpretar mejor los resultados que obtuvimos.

Modelo KNN (k-nearest neighbor)

Para el punto (2) del trabajo, se nos dio la tarea de crear un modelo de clasificación binario que, aprendiendo de un set de entrenamiento conformado por las imágenes correspondientes a las letras A y L, logre determinar correctamente la clase a la que pertenecen futuras muestras. De esta manera, previo al desarrollo del modelo, calculamos la cantidad de imágenes de cada clase e hicimos una partición del dataset para aplicar Cross-Validation: consideramos, por un lado, que no hay una enorme diferencia cuantitativa entre las dos clases, por lo que, en consecuencia, decidimos hacer un train-test split normal; elegimos entrenar el modelo con un 70 porciento del dataset original, guardando el 30 porciento restante para la evaluación (aunque también se podría haber

hecho una partición 80/20).
Con respecto al modelo, decidimos fijar 4 combinaciones aleatorias de píxeles (atributos) para llevar a cabo la clasificación (aunque se podrían haber elegido más o menos; observamos después de probar con distintos valores que 4 reflejaba bien los intervalos de score), y elegimos como métrica la exactitud. Además, para este experimento se hizo variar k (la cantidad de vecinos) para cada una de las combinaciones, y se hizo un gráfico que plotea la exactitud del modelo (para los atributos dados) en función de los valores que toma k. Decidimos agregar uno sólo de los primeros 4 al informe porque consideramos que es suficiente para comentar lo que consideramos pertinente (aunque incluimos el resto en el código).

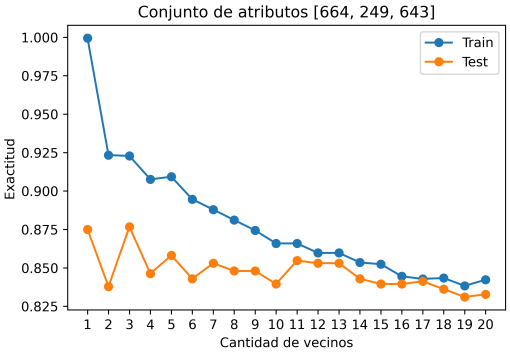

Figura 5

En el primer gráfico, observamos que la exactitud (tomada para el test set) tiene un máximo en $k = 3$, aunque es importante resaltar que $k = 1$ devuelve un score casi tan bueno o, para otras combinaciones de atributos, incluso mejor que éste. Uno se inclinaría a pensar que en este caso, 1-NN generaliza y predice mejor que cualquier otro k, pero hay que tener cuidado: si bien es posible que sea el mejor modelo, también tiene, como predictor, una mayor varianza que k-NN para mayores valores de k. Esto quiere decir que es más propenso a clasificar mal para otras particiones del dataset en set de entrenamiento y evaluación. Dado que bajo esta métrica (y esta cantidad de atributos) $k = 3$ obtiene resultados tan buenos o mejores que 1-NN, nos parece más conveniente (por las razones ya mencionadas) elegir este valor de k.
Finalmente, decidimos graficar la misma relación que en el anterior, pero esta vez cambiando la cantidad de atributos ponderados para la clasificación.

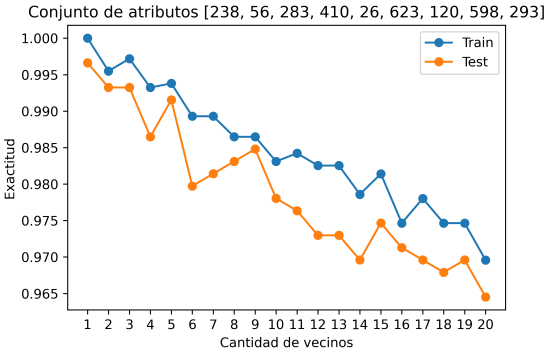

Figura 6

Notemos que ocurren dos cosas: para empezar, observamos que el modelo con más atributos obtiene mejores resultados (con respecto a esta métrica); este es un patrón que se repite en todos

los experimentos que hicimos. Además, las diferencias entre los scores del set de entrenamiento y test son menores que en el caso anterior. A partir de esto, pensamos que el modelo predice mejor a medida que incrementamos el número de píxeles. Una posible causa que podría explicar esto es la información que gana el modelo al agregar nuevos atributos, aunque no descartamos la idea de que el motivo por el que ocurre esto sea otro.

Además, pensamos en otros dos métodos posibles para esta parte del trabajo: la idea de la primera opción es agarrar una cantidad predefinida de píxeles de manera azarosa y hacer el entrenamiento del modelo con la combinación obtenida, pero esta vez restringiendo el total sobre el cual hacemos la selección a los atributos donde más difieren las dos letras. Es decir, a diferencia de lo que hicimos previamente, nos limitamos a las filas y columnas que, en promedio, mejor logran diferenciar ambas clases. Así, obtuvimos los siguientes resultados:

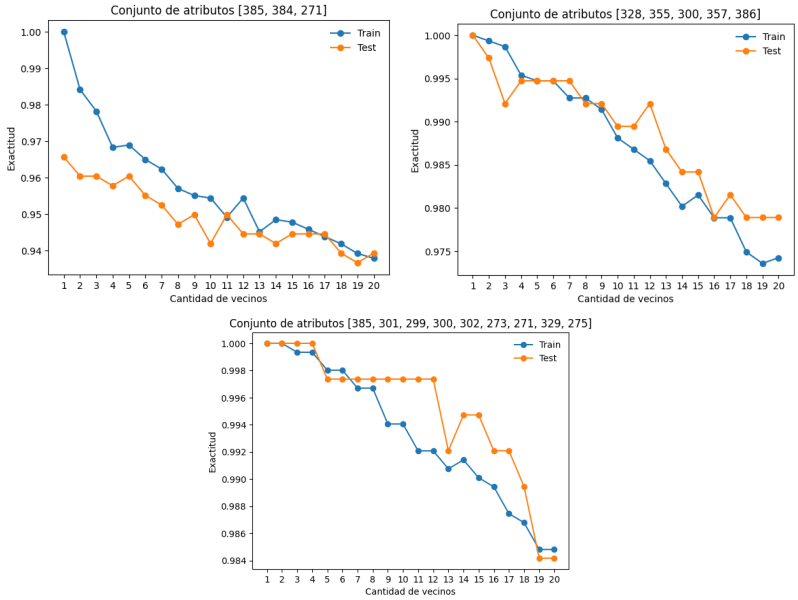

Figura 7

Se puede ver que si bien los patrones del primer método se repiten acá, los resultados parecieran ser, en líneas generales, mejores (utilizando la exactitud como métrica).

Finalmente, para el método 3 hicimos el mismo experimento, cambiando el criterio: el conjunto de píxeles sobre el cual tomamos azarosamente los atributos que necesitamos para evaluar el modelo queda conformado por aquellos que, para las imágenes promedio de las dos letras, tienen una diferencia mayor a 55 entre sí. Coloreamos a los elementos de este conjunto de negro o blanco en la siguiente imagen.

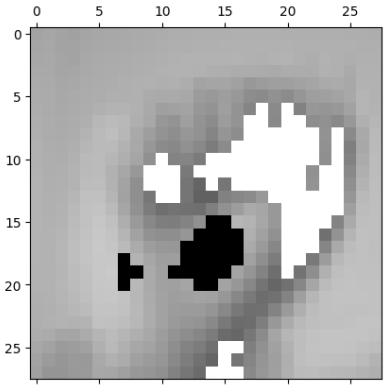

Figura 8

Se generaron modelos para 3 combinaciones de los atributos con mayor diferencia entre las clases, primero para 3 atributos, luego para5 y 9. Si bien estos modelos presentaron una performance similar a la de los modelos generados con los métodos anteriores, se pudo notar una mejoría, particularmente para el caso de los modelos con 5 atributos. Debido a esto decidimos tomar como modelo final a un modelo entrenado con los atributos [716, 271, 302, 357, 770] con k = 7 (dado que fue el valor de k que tuvo una performance similar sobre los datos de train y test - como se puede ver en la figura lll).

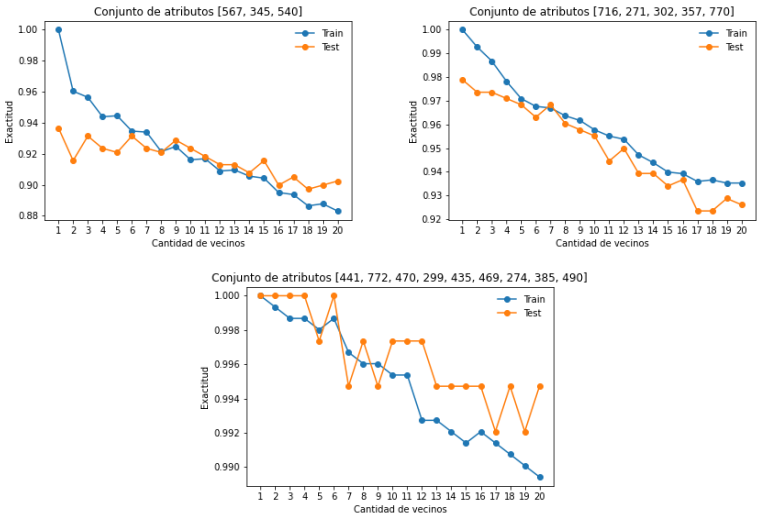

Figura 9

Decidimos entonces evaluar el modelo electo con los datos de evaluación que se separaron en un principio, lo que devolvió una performance de 0.932. Si bien la exactitud del modelo es ligeramente menor que cuando se evaluó con los datos de desarrollo, consideramos que de todas formas el modelo es apropiado para predecir con datos nuevos.

Modelo de arboles de decisiones

Por otro lado, para el punto (3) del trabajo se deseaba averiguar a qué vocal corresponde la seña en una imagen. Para esto se tomó un subset de datos que contuviesen solo las clases de las vocales (A, E, I, O, U). El mismo contenía una distribución armónica de datos de cada clase, como se puede observar en la figura 7.

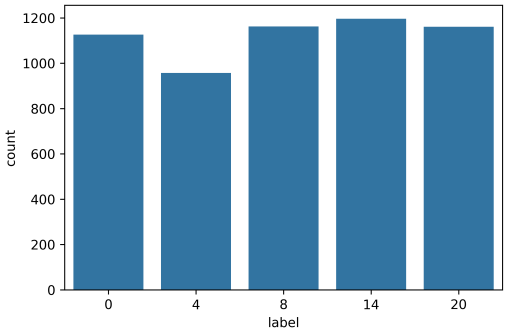

Figura 10: Distribución de datos para las clases A (0), E(4), I(8), O(14) y U(20). Si bien la E cuenta con menos datos que el resto ésto se considera despreciable dada la cantidad de datos con la que se cuenta.

Para poder determinar qué vocal corresponde a una imagen se implementó un modelo de aprendizaje supervisado llamado árbol de decisiones, con el fin de evaluar y comparar distintos modelos creados con el mismo mediante el método de validación cruzada con k-folding y otras métricas para clasificación multiclase. En principio se realizó una división del subset para datos de entrenamiento y datos de evaluación. Luego se generaron distintos árboles de decisiones variando su altura desde 1 hasta 10 y se evaluó la exactitud de cada uno, como se ve en la figura 8.

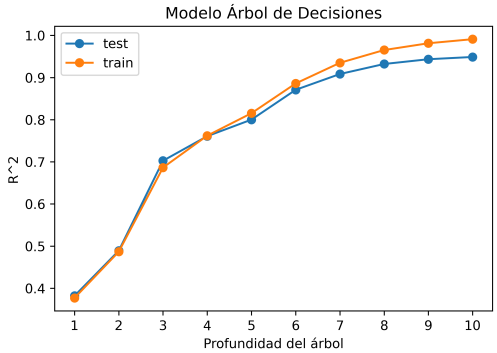

Figura 11: Altura de un modelo de árbol de decisiones vs la exactitud de su performance en los datos de training y datos de testing, respectivamente.

Se detectó que la performance de los modelos de árbol era similar entre los datos de train y testing para las alturas 1 a 6. Sin embargo, para las alturas 1, 2 y 3 el valor de exactitud es muy bajo, por lo que el modelo tiende a subajustar para esos valores. A partir de la altura 7 la exactitud sobre los datos de training y sobre los datos de testing se alejan, por lo que a partir de esa profundidad el modelo tiende a sobreajustar, lo que nos lleva a concluir que a un buen modelo de árbol de decisiones para este problema le corresponde un valor de altura entre 4 y 6. Por esto se generaron 3 modelos de árboles nuevos, cada uno con una altura de 4, 5 y 6 respectivamente, a los cuáles se los evaluó aplicando k-folding con validación cruzada. Para el k-folding se determinó un $k = 10$, considerando que se contaba con un set de training de aproximadamente 3500 imágenes. A su vez, se evaluó a cada uno de los 3 modelos analizando su precisión y su exhaustividad (o recall). Adicionalmente, se generó para cada modelo una matriz de confusión (figura 9), para apreciar mejor la certeza de sus respectivas predicciones.

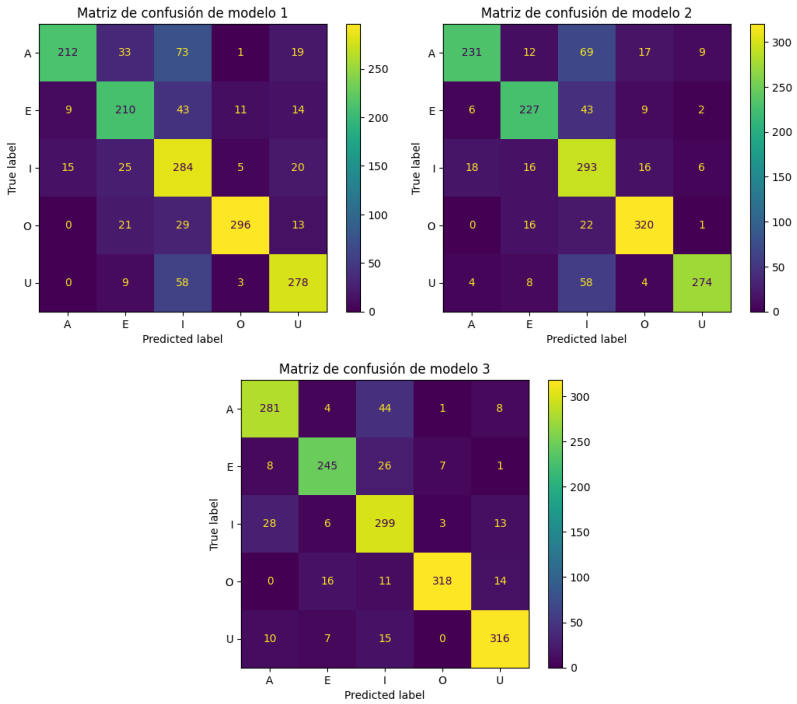

Figura 12: Matrices de confusión para modelos de árbol de decisiones con altura = 4 (derecha superior), 5 (izquierda superior) y 6 (abajo). Se puede apreciar que el último presenta mejor precision y recall.

Hay varias cosas a tener en cuenta con respecto a las matrices de confusión: por un lado, vemos que los primeros dos modelos tienen una mayor tendencia a asignar muestras correspondientes a las letras A, E y U a la clase I, problema que pareciera corregirse parcialmente en el modelo 3. Además, comparando las matrices 1 y 2, vemos que los errores de clasificación son bastante parecidos en ambos casos, aunque el segundo modelo pareciera ser el mejor de los dos, guíandonos por los valores que toman sus matrices en la diagonal (y después confirmándolo usando los resultados de las métricas que conocemos). Asimismo, creemos que el tercer modelo es el mejor de los tres, viendo que es menos propenso a cometer los errores de clasificación ya mencionados, y considerando que es el que más se asemeja a una matriz de confusión ideal.

3. Conclusiones

Luego de generar modelos de predicción con los algoritmos KNN y árboles de decisiones y analizar su performance, se llegó a la conclusión de que el modelo K-nearest neighbor tiende a sobreajustar cuando se le provisionan pocos atributos y devuelve mejores resultados con grandes cantidades de atributos, posiblemente gracias a la información que gana al hacer esto. Respecto al modelo de árboles de decisiones, se determinó, a través de numerosas métricas para clasificación multiclase como la precisión, recall, matrices de confusión y cross validación junto con k-folding, que al modelo que mejor ajusta los datos le corresponde una altura de 6, pues devolvió los mejores resultados en todas las métricas mencionadas previamente. Se concluye entonces que se pueden implementar modelos de aprendizaje supervisado para reconocer lenguaje de señas en imágenes.