DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo practico 02

10 de marzo de 2024 Laboratorio de Datos

pip install grupo_1

Integrante	LU	Correo electrónico
Rocca, Santiago	152/23	santiagorocca17@gmail.com
Moguilevsky, Agustin	951/23	agumogui@gmail.com
Pina, Martin	320/23	martinpina04@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

Introducción:

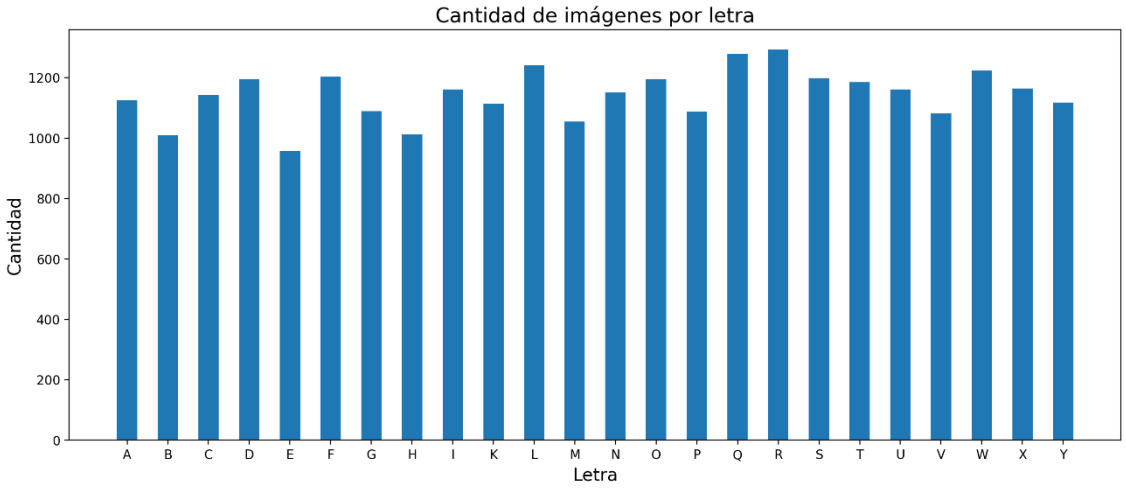
En este trabajo se realizaron implementaciones de modelos de calificación, tanto binarios cómo multiclase, donde se crearon modelos de predicción para diferenciar letras en lenguaje de señas.

Análisis Exploratorio de Datos:

Llevamos a cabo el trabajo a partir del conjunto de datos "Sign Language MNIST". Este contiene imágenes de señas del abecedario. Es importante destacar que no todas las letras se encuentran en este dataset, ya que las letras 'z' y 'j' no se realizan mediante señas, si no a través de gestos. Esta base de datos tiene una etiqueta ('Label') que indica qué letra se está representando, seguida de los valores de los píxeles de la imagen. Cada imagen posee 784 píxeles. Construyendo una matriz de 28x28 y convirtiéndola en un mapa de calor, obtenemos la imagen de cada seña.

Muestras:

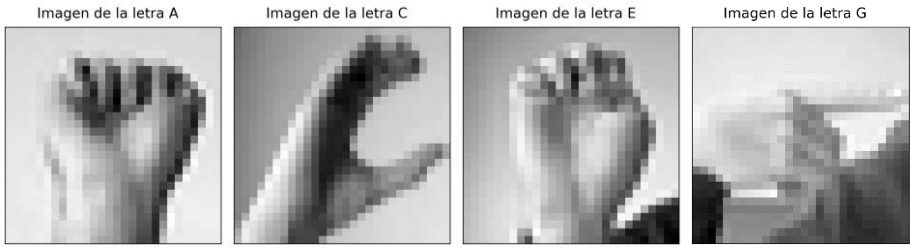
En primer lugar, buscamos responder si las muestras por letra eran proporcionales. Contamos la aparición de cada 'Label' utilizando consultas SQL obteniendo el siguiente resultado:



Como se puede observar en el gráfico la cantidad de muestras es proporcionada. El rango de muestras se mueve entre las 957 y 1294 imágenes por letra.

Primeras Imágenes y Pixeles Importantes:

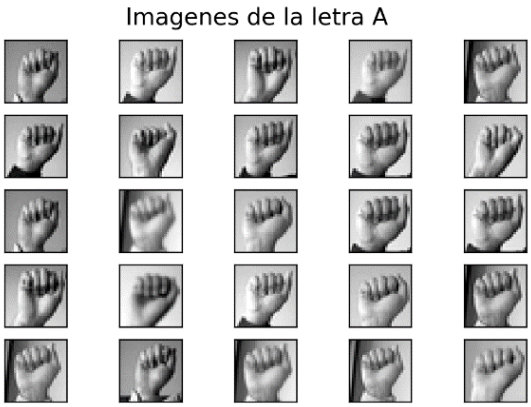
Recreando las primeras imágenes, observamos ciertas similitudes entre todas ellas. Las señas suelen encontrarse principalmente en el centro de la imagen, mientras que en el fondo se aprecian diferentes tonos que distinguen y destacan la forma de estas. Es por ello por lo que clasificamos como píxeles importantes aquellos que representan una parte de la mano, mientras que consideramos irrelevantes aquellos que representan el fondo. Por lo tanto, si tuviéramos que descartar píxeles, serían estos últimos.



Similitud Entre Señas:

Debido a la escasez de píxeles en las imágenes, a veces resulta difícil distinguir una señal de otra, especialmente cuando implican una posición similar de la mano y/o los dedos. Por ejemplo, es complicado diferenciar la letra A de la E o la letra M de la N.

Por esta razón, hemos desarrollado un método para identificar los píxeles importantes que nos permiten distinguir entre una letra y otra. Una ventaja de este conjunto de datos es que las imágenes de cada letra tienden a ser similares. Por ejemplo, aquí tenemos 25 imágenes de las letras A y L en las cuales podemos observar similitudes como las sombras, la inclinación de la mano y la posición de la muñeca.

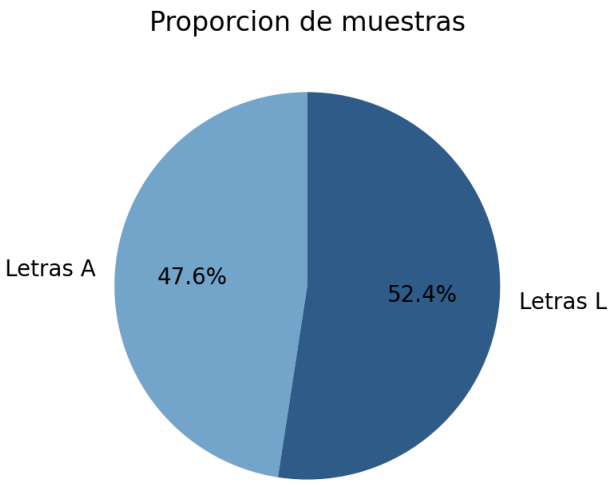


Basándonos en el análisis previo, consideramos que la exploración de los datos puede ser más complicada en comparación con el conjunto de datos del Titanic visto en clase. Sin embargo, con las herramientas adecuadas y representando los datos en un mapa de calor, podemos llevar a cabo el trabajo sin ningún problema.

Experimento Modelo AoL:

Proporción de muestras, conjunto de entrenamiento y conjunto de prueba:

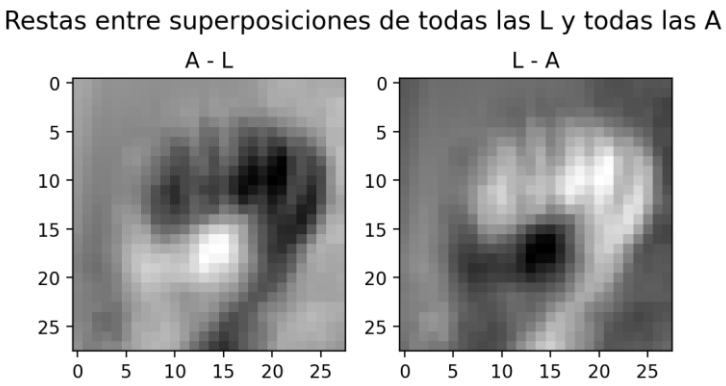
Con este experimento queremos crear modelos KNN (K Nearest Neighbor) para predecir, a partir de una imagen, si se trata de una letra “A” o “L”. En primer lugar, obtuvimos todas las muestras correspondientes en un dataset llamado “Muestras AL” que contenía las letras de interés y conformamos este gráfico para visualizar la cantidad de estas.



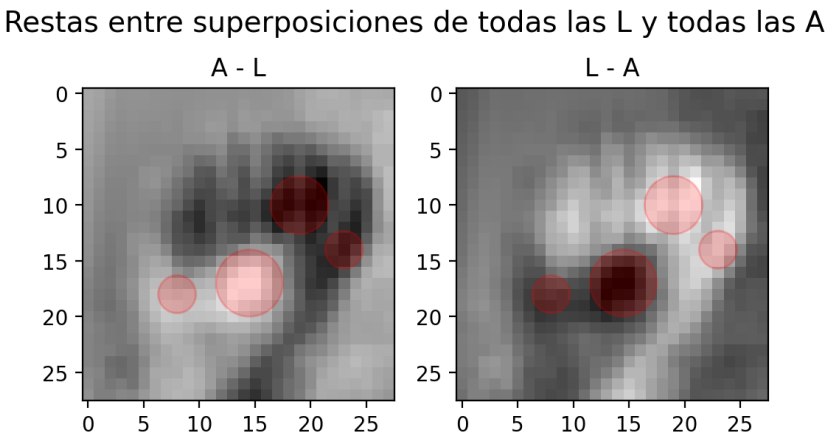
Podemos observar en este gráfico que las muestras dentro de este data frame están equilibradas, por lo que la disparidad en la cantidad de "A" y "L" no será un problema. Dividimos este conjunto en dos sets: uno de entrenamiento y otro de prueba para evaluar la eficacia del modelo y verificar si cumple nuestras expectativas. El set de prueba representa el 20% de la muestra original.

Superposición de Imágenes y entrenamiento del modelo:

Para determinar qué píxeles son importantes superpusimos todas las imágenes correspondientes a cada letra para formar una única imagen combinada. Luego, restamos las matrices, obteniendo el siguiente resultado:



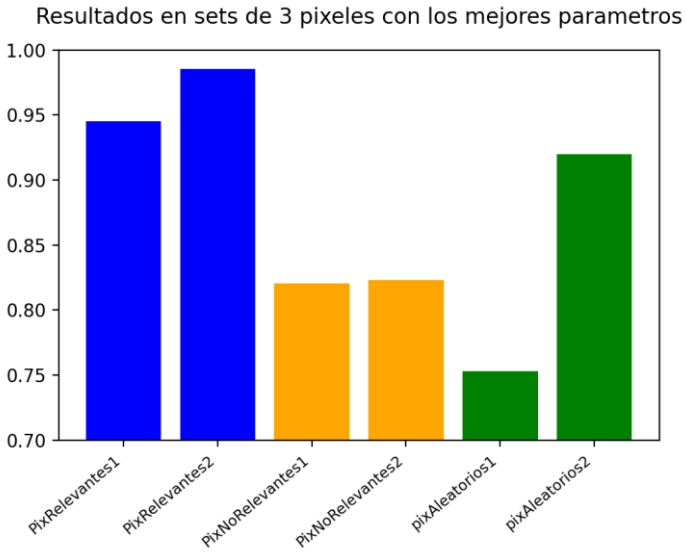
De esta manera, identificamos los píxeles con mayor variabilidad, los cuales se muestran como los más blancos y negros:



Entrenar un modelo utilizando estos píxeles consideramos que será más eficiente y simplificará la detección de la letra correspondiente. Por lo tanto, tomando conjuntos de tres pixeles relevantes ({491,492,518} , {301,274,246}), irrelevantes ({283,784,10} , {1,420,28}) y aleatorios ({448,462,599} , {648,320,187}) entrenamos el modelo.

Predicciones del Modelo:

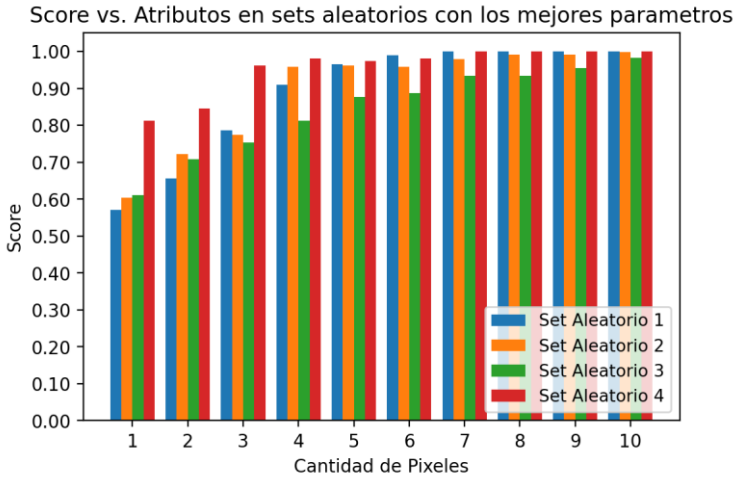
Evaluamos la exactitud del modelo en base a cada conjunto utilizando las muestras de prueba y estos fueron los resultados:



A partir de esta imagen podemos concluir que existe una relación en la importancia de la elección de píxeles para obtener un mejor *score*. Sin embargo, el modelo automático con píxeles no relevantes obtiene un buen resultado.

Score vs Cantidad de Pixeles:

En base a este modelo automático, estudiamos cómo se comportaba con más atributos. Generamos 4 sets aleatorios de un único píxel, los cuales entrenamos y evaluamos su exactitud. Por cada píxel agregado a cada set, repetimos el proceso de evaluación y visualizamos el resultado en el siguiente gráfico:

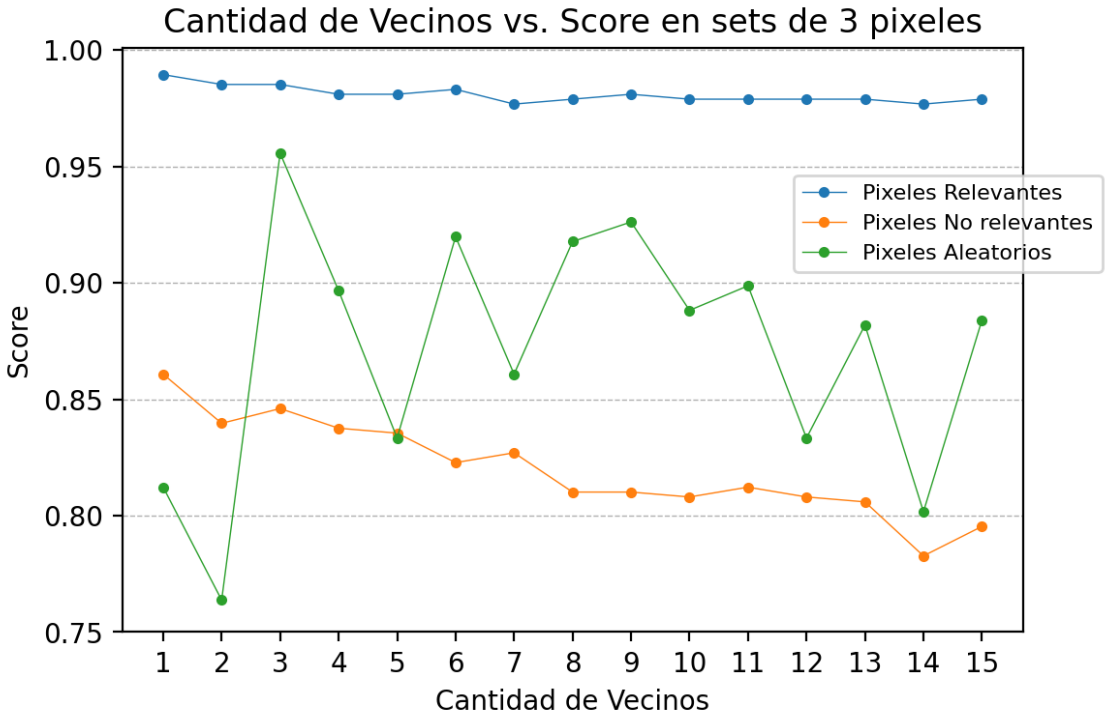


Se puede observar que existe un patrón en donde el modelo predice mejor cuando posee mayor cantidad de píxeles. Esto nos llevó a generar el siguiente modelo no automático.

Modelo AoL No Automático:

Score vs Cantidad de Vecinos:

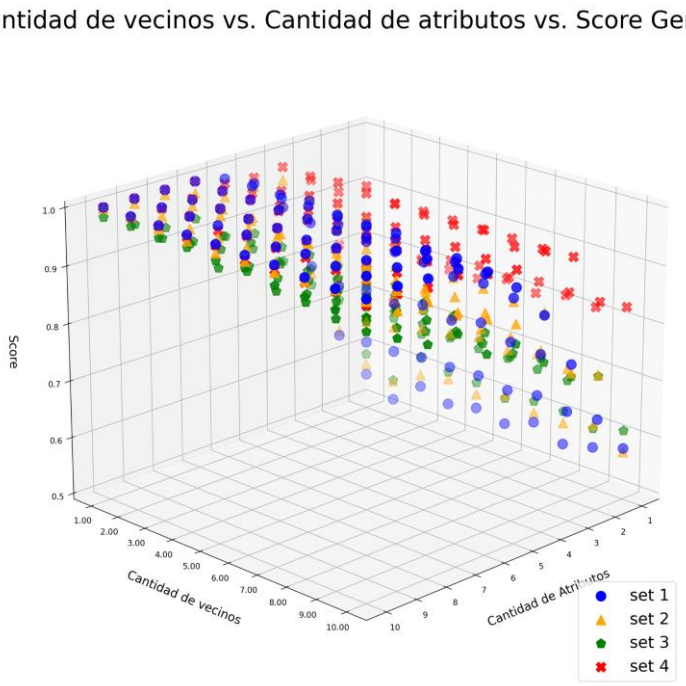
Desarrollamos un modelo KNN en donde se toma como entrada la cantidad de vecinos y los píxeles de interés para el usuario. Utilizando 3 de los conjuntos mencionados previamente (uno por cada condición de relevancia), fuimos evaluando su exactitud a medida que aumentamos la cantidad de vecinos. A continuación, el gráfico correspondiente a ese proceso:



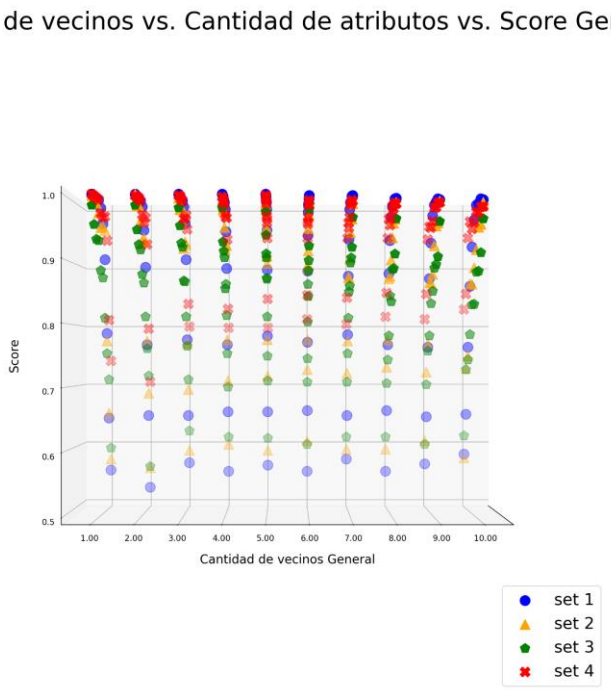
A medida que el modelo evalúa con mayor cantidad de vecinos obtiene un resultado inferior y el mejor score se da cuando toma k = 1 en los tres casos.

Score vs Cantidad de Atributos vs Score General:

Por último, evaluamos el score en función de la cantidad de píxeles y vecinos. Generamos 4 sets de píxeles aleatorios. A medida que le agregamos atributos, calculamos su precisión con distinta cantidad de vecinos:

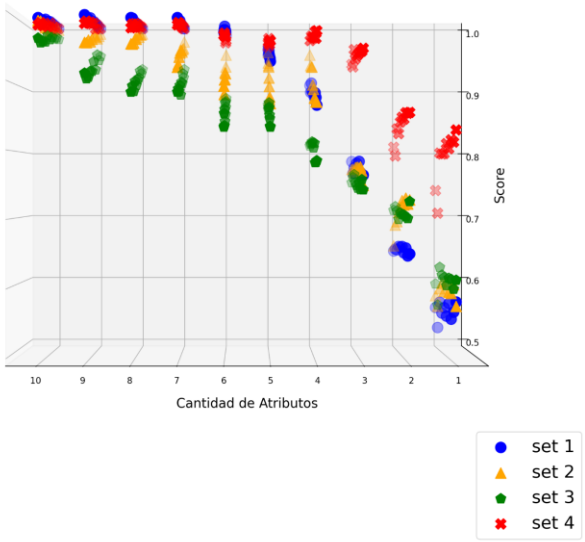


En esta perspectiva no se puede apreciar una relación clara entre las variables. Cambiando la perspectiva:



No se aprecia alguna correlación clara entre el score y la cantidad de vecinos, pues para un mismo valor de k tenemos muy variados resultados de score. Sin embargo, cuando cambiamos nuevamente la perspectiva:

Cantidad de vecinos vs. Cantidad de atributos vs. Score General



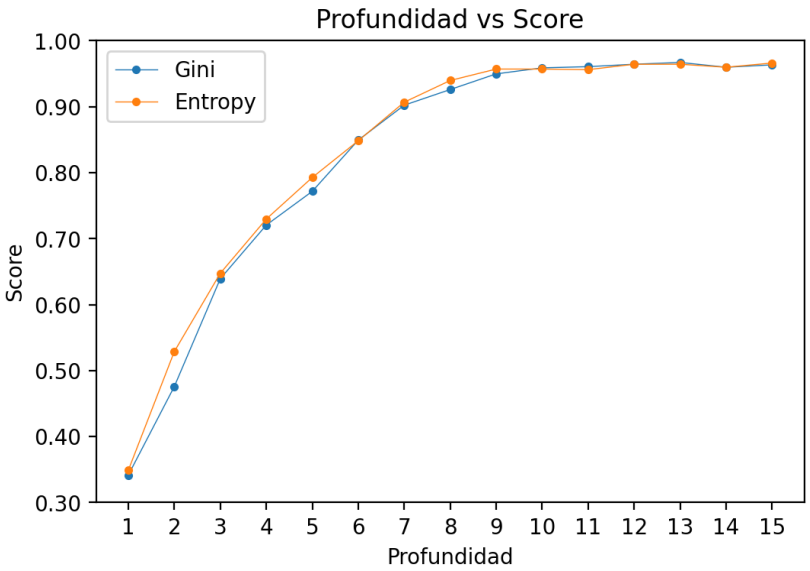
Se observa que cuantos más píxeles se utilicen, sin importar su relevancia, se obtendrá un score más alto y por lo tanto será más fácil detectar si se trata de una A o una L.

Experimento Modelo Vocales:

Data Frame, Conjunto de Entrenamiento, Conjunto de Prueba:

En este experimento, buscamos predecir a qué vocal hace referencia una imagen utilizando un modelo predictivo de árbol de decisión para clasificación multiclase. Para ello, primero reunimos todas las muestras de todas las vocales en un solo data frame. Dividimos esta muestra en un 80% para el entrenamiento del modelo y el restante 20% como conjunto de prueba para evaluar el rendimiento del modelo.

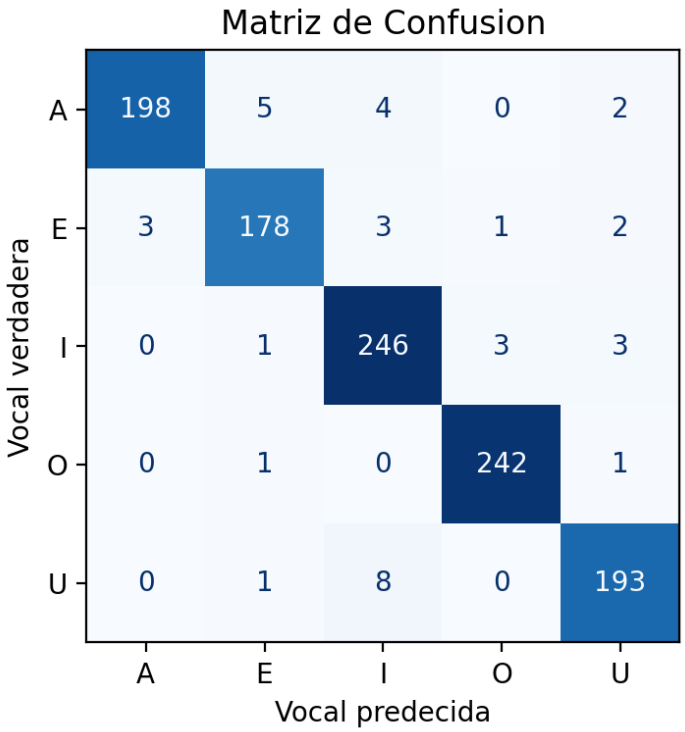
En un primer paso, evaluamos este modelo con varias profundidades y obtuvimos los siguientes resultados:



Se puede observar que a medida que hay mayor profundidad se obtiene un mejor score.

K-fold-cross-validation y Matriz de Confusión:

Para comparar y seleccionar los árboles de decisión utilizamos k fold cross validation. Los hiper parámetros utilizados fueron Gini y Entropy como criterios y las alturas desde 1 a 15. Optamos por utilizar GridSearchCV, y el resultado obtenido fue un árbol de profundidad 11 con criterio Gini. A partir de los datos, generamos la siguiente matriz de confusión:



Los resultados obtenidos en la matriz revelan otra imagen de la exactitud demostrada anteriormente, ya que en pocas ocasiones el modelo confunde las vocales, esto se asemeja al buen score de exactitud obtenido anteriormente con profundidades a partir de 10.

Conclusiones:

Una peculiaridad de los modelos KNN generados con sets de 3 pixeles es que el mejor hiper parámetro siempre es 1 único vecino cercano.

El modelo de clasificación binaria nos dio como resultado que a medida que aumentan los atributos mejora el rendimiento del modelo.

No existe una relación clara entre score y cantidad de vecinos, la relación predictiva más directa está entre score y cantidad de atributos.

En la clasificación multiclase a medida que hay una cantidad considerable de profundidad no se genera overfitting.