Trabajo Práctico 2

Grupo Abeliano

Integrantes:

- Moyano Tassara, Iván; LU: 237/21; Correo: ivanmoyano@hotmail.com
- Zubieta Hernandez, Sebastian; LU: 209/23; Correo: sebastianzubieta04@gmail.com
- Galli Casado Sastre, Lucas; LU: 739/21; Correo: lucasgalli01@gmail.com

Introduccion

Una de las primeras observaciones que hacemos al ver el dataframe con el que trabajamos es que al tener 784 atributos, resulta difícil analizarlo en su estado inicial, tanto visual como analiticamente. Por esto resulta de vital importancia distinguir cuáles de estos atributos (en este caso particular pixeles) resultan ser los más significativos, a continuación describimos los métodos que utilizamos para lograr ese fin.

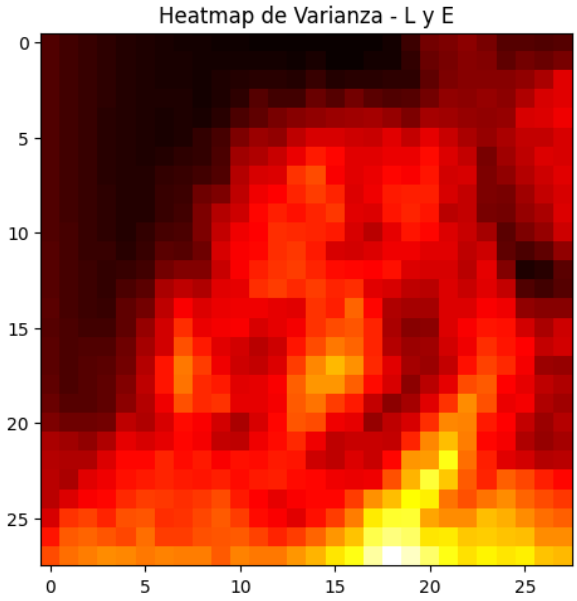
Primero buscamos reducir los datos a tres dimensiones utilizando PCA. Esto se hace mediante la definición de Z_1, Z_2, Z_3, donde cada Z_i es una combinación lineal de los atributos originales con coeficientes que son definidos con el fin de maximizar varianza y lograr ortogonalidad entre cada una de las variables nuevas.

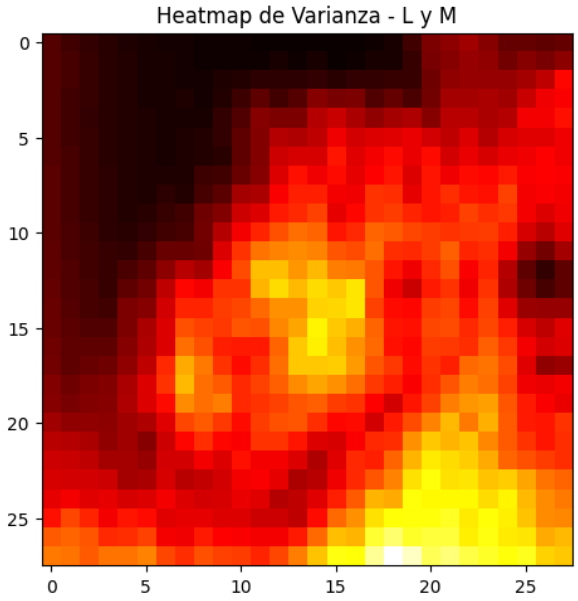
Para definir cuál de los 784 atributos de nuestros datos son los tres más relevantes, tomaremos un $1 \leq i_m \leq 784$ tal que:

$\phi_{i_m} = \max_{1 \leq j \leq 784} \phi_{mj} \quad \forall 1 \leq m \leq M$, con M = 3 en este caso particular

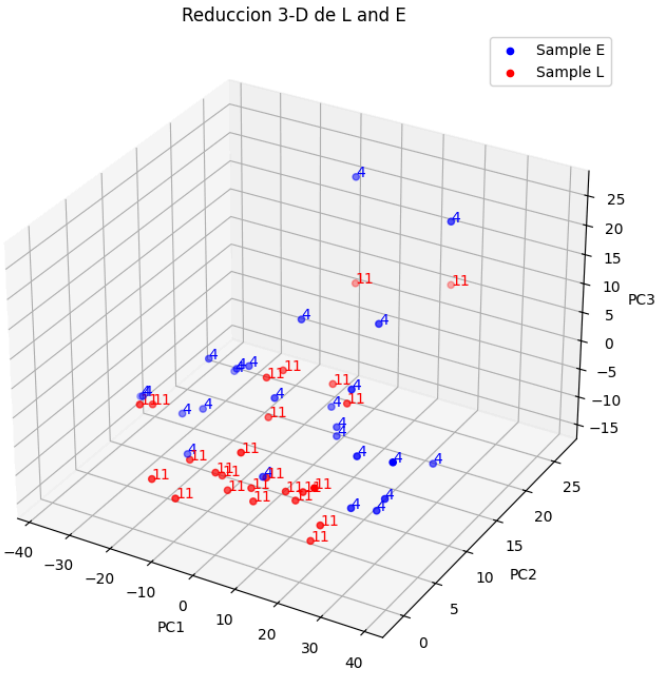
La elección de atributos se basa en el hecho que los ϕ_{ij} son hallados de forma tal que maximicen la varianza de Z_m y que las PC's sean ortogonales, por lo que los i_m que tomamos representan el elemento más significativo de cada una de los tres elementos de la base que define el espacio de dimensión M (en este caso 3) en el que plasmamos los datos. Este criterio nos da 3 atributos $X_{i_1}, X_{i_2}, X_{i_3}$. Aplicando este criterio, los píxeles más significativos son los 263, 505, y 784. Más adelante estos 3 pixeles serán utilizados en un modelo.

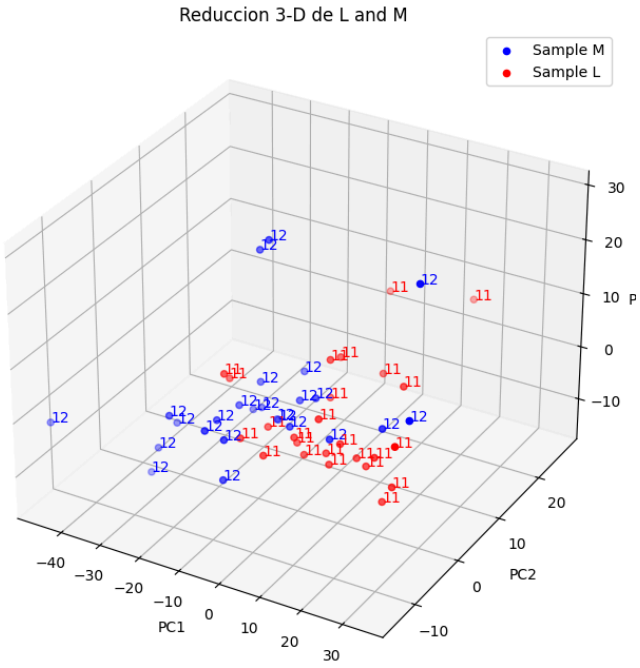
Para responder a la pregunta “¿Es más fácil diferenciar la seña E de la L o la E de la M?”, primero armamos dos datasets: El primero con todas las instancias de las señas E y M, y el segundo con todas las instancias de las señas E y L. Luego graficamos heatmaps de cada dataset, en donde cada cuadrado en la grilla de 28x28 representa la varianza del píxel de la misma posición(a mayor claridad, mayor varianza). Estos fueron los resultados: En base a estos gráficos, conjeturamos que es más fácil distinguir una L de una M que una L de una E





pues en su gráfico se observa más varianza sobre todo en el centro de las imágenes. Con el fin de verificar o no la conjetura a partir de un criterio distinto utilizamos PCA para reducir los data sets utilizados a 3 dimensiones, luego tomamos 25 puntos aleatorios de cada letra y los graficamos:





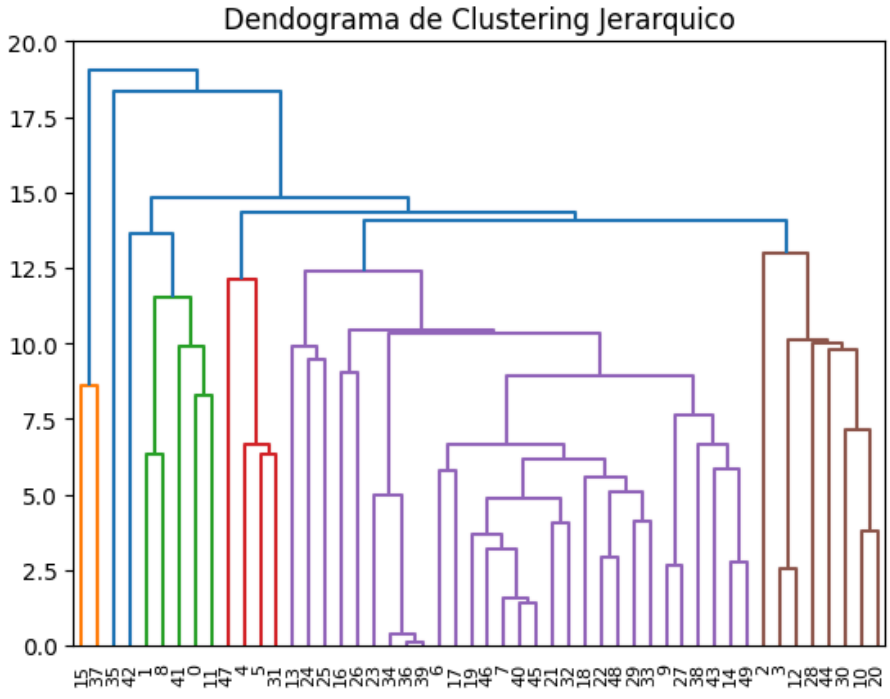
Al implementar una reducción de dimensión y además plotear pocos puntos de forma

aleatoria,hay mucha información que no está siendo visualizada, sin embargo, en este limitado contexto el gráfico parece validar nuestra conjetura.

Dada una letra, queremos saber, “¿Son todas las imágenes muy similares entre sí?”, tomamos la clase de las señas que corresponden a la letra C para evaluar realizamos lo siguiente:

Primero tomamos una muestra aleatoria de nueve señas de la letra C para poder evaluarlas nosotros. Las imágenes que vimos fueron las siguientes:

A simple vista se observa que las principales diferencias se pueden encontrar en el posicionamiento del pulgar y el ángulo en el que se posicionan los dedos. Para el ojo resulta fácil filtrar estas diferencias y observar la letra que representan, pero queremos ver si esto sigue siendo el caso si no está la opción de visualizarlas de esta manera. Para aplicar un criterio más empírico utilizamos clustering jerárquico y un dendograma. Queremos observar a partir de qué distancias podemos observar un solo cluster, ya que sabemos de antemano que todos los datos sobre los que aplicaremos el algoritmo son instancias que corresponden a la letra C.



Dado el alto costo computacional del clustering jerárquico decidimos aplicarlo sobre una muestra aleatoria de 50 instancias de la letra C.En este ejemplo en particular, la máxima distancia entre dos puntos es más de 79 y recién con epsilon mayor a 17.5 tenemos un solo cluster. Este tipo de clustering es sensible a outliers, lo cual podría ocasionar este problema , sin embargo, para llegar a pocos cluster se debe tomar el epsilon grande. Decimos “grande” pues es significativo en proporción a la máxima distancia entre dos datos.

A ojo las imágenes resultan razonablemente parecidas, pero el aplicar clustering jerárquico nos brinda un resultado diferente. La semilla utilizada para todas las selecciones random fue 457.

Creemos que sí se complica la exploración de datos al manejar tantas variables, ya que si bien luego de realizar una reducción de dimensión utilizando el método PCA, ya no hay ningún problema, se pierde mucha información al utilizar este método y resulta costoso.

Experimentos

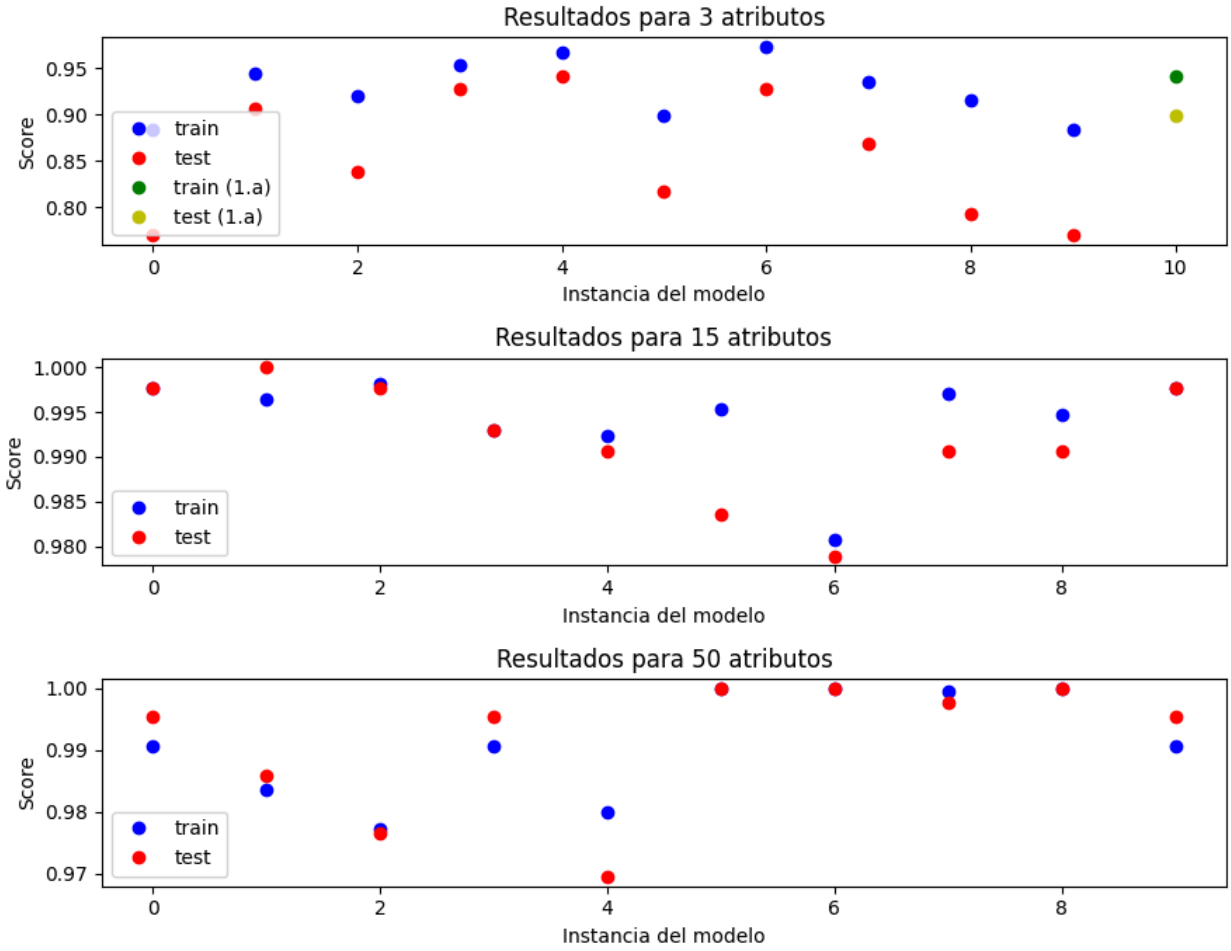
KNN - Clasificación de L y A

El primer paso fue crear una base de datos con solamente las instancias correspondientes a las letras L y A, tras hacer esto separamos un 10% de los datos para aplicar Hold-Out luego de definir los modelos definitivos que usamos. Finalmente definimos las variables dependientes y a predecir(labels) y aplicamos un training-test split(80% y 20% resp.) sobre ellas.

Para 30 de los modelos implementados se seleccionaron 3,15, y 50 píxeles de forma aleatoria, 10 veces para cada cifra, y luego se plantearon modelos KNN pasados en esos mismos atributos seleccionados al azar con semilla “457”. Adicionalmente se implementó un modelo basado en los 3 píxeles obtenidos a partir del ítem 1.a para ver si este criterio de importancia resulta útil a la hora de decidir qué atributos utilizar para un modelo.

Tras plantear los modelos randomizados con 3, 15 y 50 píxeles aleatorios pudimos ver que el modelo basado en el criterio aplicado en el ítem 1.a resulta en un mejor rendimiento. El modelo que utiliza los 3 píxeles obtenidos anteriormente(263, 505, y 784) obtuvo un score de 0.869 al evaluarlo con Hold-out, mientras que el mejor modelo aleatorio de 3 pixeles obtuvo un score de 0.806 y el mejor modelo aleatorio de 15 pixeles obtuvo un score de 0.810 (ambos calculados con Hold-out). Al implementar un modelo aleatorio de 50 píxeles el rendimiento del modelo propuesto por significación fue opacado ya que el modelo de 50 píxeles aleatorios logró un score de 0.983 al evaluarlo con Hold-out.

Los modelos que se consideraron para llevar a la instancia de hold-out fueron el 6to modelo aleatorio con 3 píxeles, el 2ndo modelo aleatorio con 15 píxeles, el 5to modelo aleatorio con 50 pixeles y el modelo de 3 pixeles propuesto por los resultados análisis del 1.a. Los modelos aleatorios fueron seleccionados a partir del siguiente gráfico:



Los puntos azules y rojos corresponden a instancias de modelos aleatorios mientras que los puntos verdes y dorados corresponden al modelo hecho a partir del ítem 1.a. Los score fueron medidos con un training-test split y la random seed utilizada fue 457 y los píxeles correspondientes a los modelos aleatorios implementados pueden encontrarse en “atributosXmodelo”.

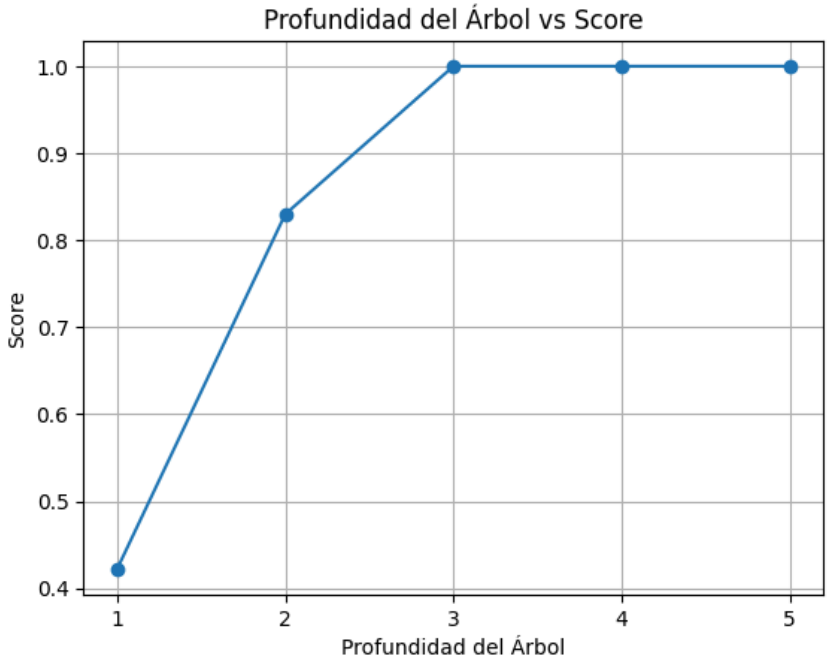
Clasificación de vocales

Para realizar una clasificación multiclase de las vocales primero se realizó un filtrado del archivo sign_mnist_train.csv en que solo solo nos quedamos con las vocales. En el mencionado archivo estas tenían como labels 0, 4, 8, 14, 20 corresponden a, e, i, o, u respectivamente.

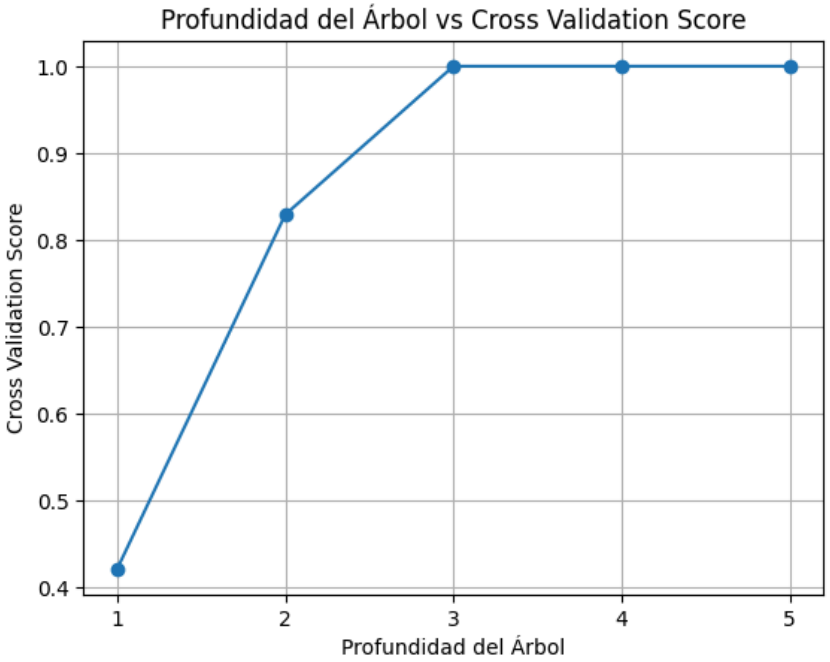
Luego se procedió con la separación de toda la data entre el train (70%), el test (15%) y el eval (15%). Este último se va a utilizar para la evaluación final, según el método hold-out.

Una vez que contamos con todos los datos requeridos y separados apropiadamente procedemos a entrenar un modelo de tree con varias profundidades, en este caso estas van

desde 1 hasta 5. Seguido de esto, se le midió el score de cada modelo de variada profundidad según la métrica de clasificación Acurracy, lo cual nos arroja el siguiente gráfico:



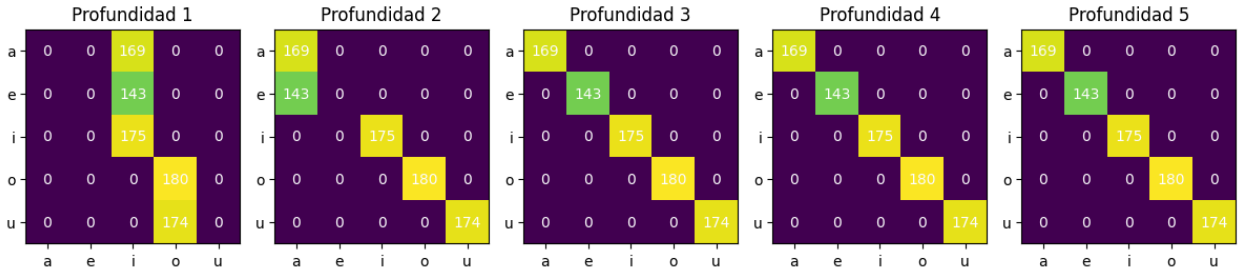
Como método adicional para la comparación de modelos utilizamos k-folding, para el cual dividimos el data training en 4 folds, y luego promediamos el score (Acurracy) que cada modelo arroja en los 4 folds. Si realizamos lo siguiente obtenemos el siguiente gráfico:



Cabe aclarar que la razón por la que se utilizaron modelos de una profundidad máxima de solamente 5 fue ya que el overfitting no se alcanzó en profundidades realistas, osea que uno

esperaría que llegado a cierta profundidad la curva de los scores empiece a bajar debido al overfitting, punto al que no se llegó con las profundidades testeadas, por lo que solo nos quedamos con trees de profundidad máxima de 5.
Otra métrica para la evaluación multiclase con la que contamos es una matriz de confusión, la cual para cada modelo nos dice cuantas imágenes clasificó bien y cuántas no, además de a cual grupo fueron erróneamente asignados, en este caso vocales:

Matriz de confusión por profundidad



Conclusiones

Clasificación de L y A con KNN

Podemos ver que el criterio definido en el 1.a que utiliza PCA y selecciona un pixel por componente principal resultó en un modelo que rinde mejor que algunos de sus contrapartes aleatorios con mayor cantidad de píxeles. Sin embargo, al aumentar la cantidad de píxeles de forma muy significativa, como fue el caso de pasar de utilizar 3px a utilizar 50px, el modelo aleatorio obtiene un mejor rendimiento. Vale destacar que cuando nos referimos a rendimiento de un modelo aludimos a la Accuracy del modelo medido con Hold-out.

Clasificación de vocales

Luego de observados los dos primeros gráficos sobre la clasificación de vocales queda claro como los modelos, pasada la profundidad 3, alcanzan un notable pico de performance, osea que los modelos de profundidad 3, 4 y 5 serían los candidatos para ser nuestros modelo elegido.

Haciendo uso de las matrices de confusión podemos sacar las siguientes conclusiones: Con el modelo de profundidad 1 podemos ver que se confunden las señas para la "i" con las de "a" y "e" de forma equitativa, con el modelo de profundidad 2 observamos que se confunde a la

letra "a" con la "e", otra vez de forma equitativa, y de ahí en adelante los modelos de mayor profundidad se comportan de forma óptima.

Siguiendo el principio de la navaja de Ockham, nos quedamos con un tree de profundidad 3 como nuestro mejor modelo para la predicción de vocales en lenguaje de señas. Para obtener el score definitivo de este modelo hacemos uso del data ser eval, el cual nos da un score de 1.