UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Clasificación y validación cruzada

Segundo trabajo práctico para la asignatura “Laboratorio de Datos” impartida el primer cuatrimestre del año 2023

Grupo “Pythonisos”

Integrantes: Nicolás Rozenberg, Joaquín Viera, Luca Zanela

Índice

Introducción

En este informe presentaremos los resultados obtenidos al trabajar con la clasificación y selección de modelos utilizando validación cruzada, utilizando el conjunto de datos MNIST [1], que incluye datos que representan imágenes de dígitos entre ***0*** y ***9***, manuscritos. Para comenzar, realizamos un análisis exploratorio de los datos. Luego, evaluamos los rendimientos de distintos modelos con distintos parámetros sobre los conjuntos de datos provistos. El primer modelo considerado fue el kNN para clasificar únicamente las clases ***0*** y ***1***. Posteriormente, utilizamos Árboles de decisión para realizar una clasificación completa (considerando todas las clases). Por último, evaluamos el mejor modelo, según nuestro criterio, para cada caso, utilizando nuevos conjuntos de datos provistos por la cátedra.

Al iniciar el análisis exploratorio, observamos que la primera columna del conjunto de datos correspondía a la clase o dígito representado por la imagen en cuestión. Las demás 784 columnas representaban los píxeles de una imagen de 28x28 píxeles, y el valor numérico indicaba la intensidad de cada píxel en escala de grises, donde 0 representaba negro y 255 blanco.

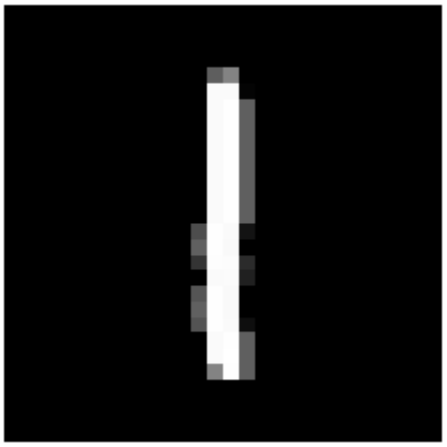

Figura 1*: Representación del dígito* ***1****, por una matriz de 28x28 píxeles.*

A continuación, contemplamos la distribución de las clases, observando que los dígitos ***1*** y ***7*** presentaban una mayor cantidad de apariciones, y el 5 una menor cantidad de apariciones. Igualmente, esta diferencia no era significativa, dado que el desvío estándar de la cantidad de apariciones de cada clase era 339, con una media de 6000, con una muestra de tamaño N=60000.

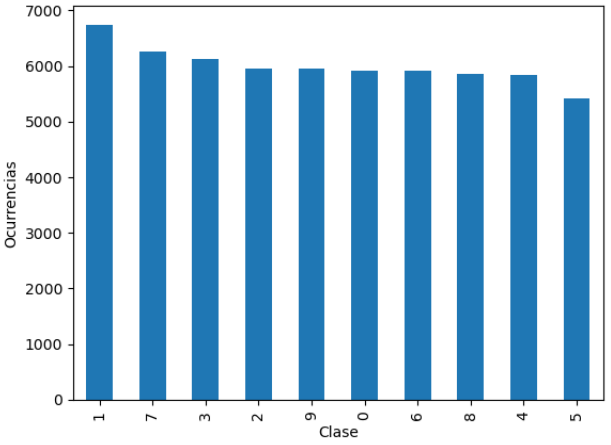

***Figura 2**: Distribución de las clases*

Luego, verificamos si los dígitos podrían ser distinguibles a partir de considerar un menor subconjunto de atributos, en lugar de todos los píxeles. Para esto, utilizamos el método PCA (Principal Component Analysis). El mismo obtiene los autovectores asociados a los autovalores mayor magnitud de la matriz de correlación de los datos, a los que denomina Componentes Principales. Con los mismos realiza una transformación lineal de los datos originales hacia un espacio de menor dimensión. Las direcciones determinadas por los autovectores generan que al transformar los datos se preserve la mayor cantidad de información posible. Al aplicar este método para 4 componentes, primero realizamos un gráfico de dispersión para visualizar qué tan separadas quedaban las clases, utilizando las primeras dos Componentes Principales (PC).

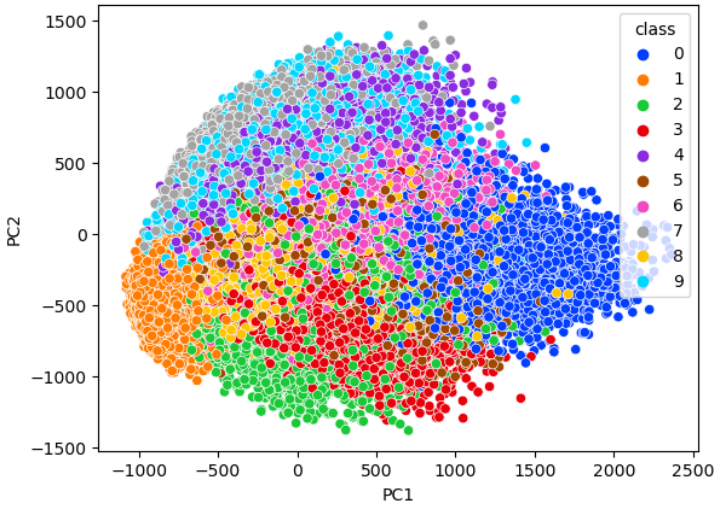

***Figura 3**: Gráfico de dispersión considerando las primeras dos Componentes Principales*

Los datos se ven solapados, lo cual tiene sentido, dado que las componentes principales graficadas solo explican el 17% de la varianza. A pesar de esto, algunos dígitos se distinguen de los demás, como el ***0***, por lo que consideramos que los datos disponibles son adecuados para realizar clasificadores de los dígitos. Luego, analizamos qué áreas de las imágenes explicaban la mayor varianza. Para esto, graficamos, para cada componente principal, un mapa de calor, donde mostramos el valor absoluto de cada píxel en la

respectiva componente principal. Esto nos dió una idea de qué tan influyente es cada píxel para determinar la dirección de cada componente principal. Esto se muestra en la Figura 4.

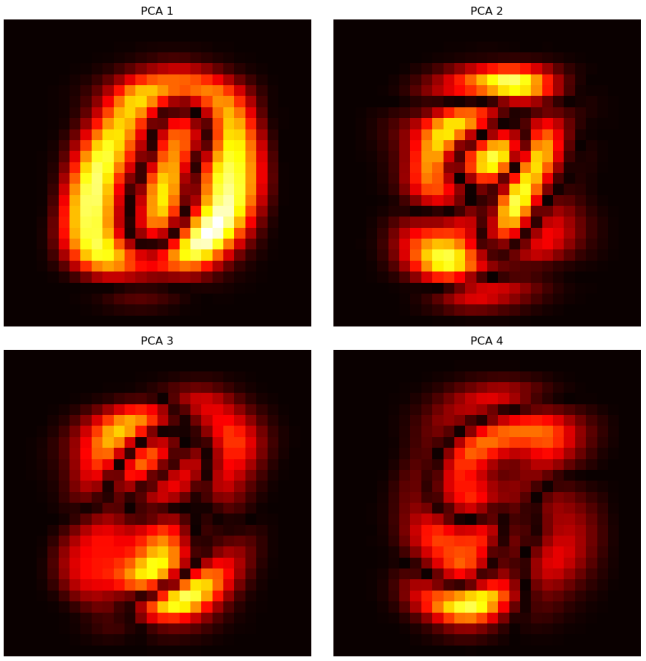

Figura 4*: Mapa de calor del valor absoluto de los píxeles dentro de cada componente principal. A mayor brillo, mayor influencia del píxel. Para obtenerlo, se enrollaron los componentes principales (vectores de tamaño 784) transformándolos a matrices de tamaño 28x28, y ponderándolos por la varianza que explicaban. Luego, se graficaron los mapas de calor de acuerdo a estas matrices.*

Observamos que los píxeles ubicados en los bordes de la imagen tienen influencia prácticamente nula en todas las componentes principales, mientras que los que se encuentran más centrados son determinantes. Notar que la zona de influencia se encuentra levemente desfasada hacia abajo, en lugar de estar centrada.

Análisis de modelos

Clasificación binaria utilizando kNN (k-Nearest Neighbours)

Para esta parte, utilizamos los datos correspondientes a las clases del ***0*** y el ***1***. Comenzamos verificando si dichas clases se encontraban balanceadas. Observamos que existe una diferencia de 6% entre la cantidad de apariciones de la clase ***1*** por sobre las de la clase ***0***, con un total de 12665 registros entre ambas clases.

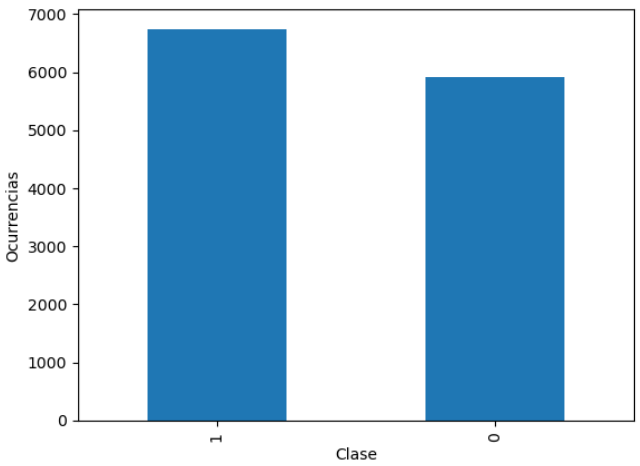

Figura 5**: Distribución de las clases **0** y **1

Comenzamos entrenando un modelo con la cantidad de vecinos k = 5, con diferentes conjuntos de atributos. Las métricas utilizadas para evaluar el modelo fueron Exactitud y F1 Score. Este último es el promedio armónico entre la precisión y la sensibilidad, donde la clase Verdadera para tomar F1 Score en este caso fue ***0***, ya que es la clase minoritaria, y por lo tanto es de nuestro interés evaluar el rendimiento para clasificarla. Por lo tanto, la sensibilidad se interpreta como la proporción de casos clasificados como ***0,*** donde la clase era realmente ***0***, y la precisión como proporción de casos donde la clase es ***0,*** dentro de los clasificados como ***0***. Preferimos darle menos importancia a la exactitud, dado que las clases - aún siendo de forma leve - están desbalanceadas de una manera que consideramos no despreciable; y asumimos de igual importancia a la precisión y a la sensibilidad. Las métricas de rendimiento para cada conjunto de atributos están reportadas en la ***Tabla 1***.

Primero, aplicamos de nuevo PCA sobre los registros pertenecientes a las clases ***0*** y ***1***, y elegimos los datos de las primeras tres componentes principales. Luego las normalizamos para que sus valores estén entre ***0*** y ***1*** (como también realizamos para los demás conjuntos de datos antes de entrenar el modelo de kNN). Luego, dividimos el conjunto de datos en un conjunto de entrenamiento y otro de validación, asignándoles una proporción del 70% y 30% respectivamente, y teniendo en cuenta el desbalance de las clases.

Luego, decidimos entrenar el modelo seleccionando únicamente los píxeles que más influencia tenían en cada componente principal, también utilizando el parámetro de cantidad de vecinos k=5.

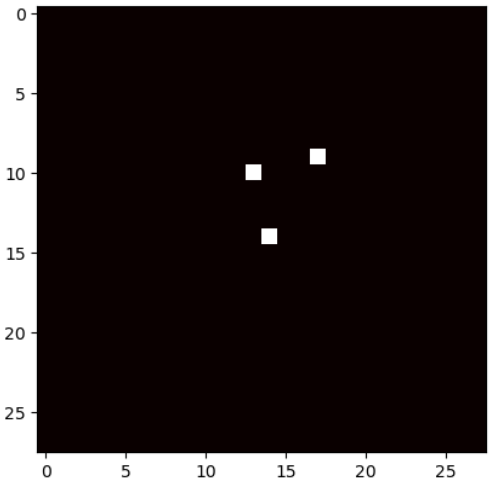

***Figura 6**: Ubicación de los píxeles de mayor influencia en cada componente principal*

A continuación, elegimos un conjunto de 25 píxeles dispuestos en forma de grilla, que nos resultaron importantes, dada su posición centrada, donde cada píxel estaba separado por píxeles de distancia tanto vertical como horizontalmente, como se puede ver en la figura 7.

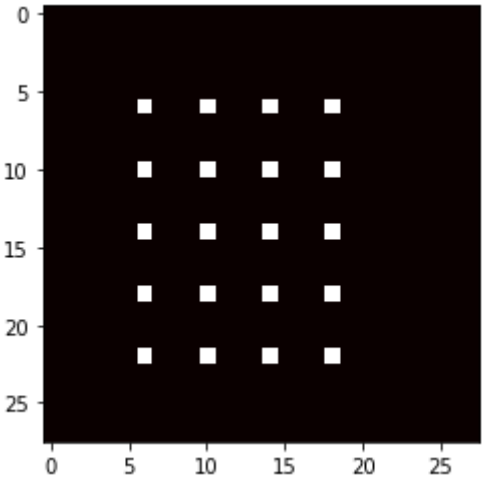

***Figura 7**: Grilla de píxeles elegidos*

De este conjunto de 25 píxeles, comenzamos eligiendo un subconjunto de cuatro de ellos de forma aleatoria, y luego creamos otros subconjuntos de cinco, seis y siete píxeles que contenían a los subconjuntos creados anteriormente de menor tamaño. Es decir, el subconjunto de seis píxeles contenía al de cinco píxeles, que a su vez contenía al de cuatro elementos. Utilizamos cada uno de estos subconjuntos de píxeles para entrenar al modelo. Esto lo realizamos utilizando una semilla de aleatoriedad para generar los mismos conjuntos cada vez que corriéramos el código. Vale notar que el rendimiento podría variar si se utilizara otra semilla. Por último, también evaluamos el rendimiento utilizando ésta grilla completa.

Variables utilizadas	Exactitud Entrenamiento	Exactitud Evaluación	F1 Score Entrenamiento	F1 Score Evaluación
PCA	0,9981	0,9971	0,9980	0,9969
Píxeles de mayor influencia (406, 269, 293)	0,9883	0,9881	0,9875	0,9873
4 píxeles aleatorios (398, 178, 510, 514)	0,9141	0,8876	0,9021	0,8701
5 píxeles aleatorios (398, 178, 510, 514, 626)	0,9142	0,8928	0,9049	0,8796
6 píxeles aleatorios (398, 178, 510, 514, 626, 186)	0,9174	0,8807	0,9083	0,8644
7 píxeles aleatorios (398, 178, 510, 514, 626, 186, 518)	0,9307	0,8986	0,9245	0,8896
Grilla Completa	0,9976	0,9968	0,9974	0,9966

Tabla 1: *Métrica de evaluación, tanto para entrenamiento como para evaluación, para el modelo kNN binario, con cantidad de vecinos k=5 utilizando distintos conjuntos de atributos. La clase Verdadera para calcular el F1 Score en este caso fue* ***0****, la clase minoritaria.*

De los resultados obtenidos, podemos notar que ninguna métrica evaluada para el conjunto de evaluación ha sido mayor que su respectivo valor para el conjunto de entrenamiento. También podemos notar que ningún valor de F1 Score ha superado su respectivo valor de exactitud. Esto se puede deber a que ***0*** es la clase minoritaria. Otro punto a observar es que, mientras que a medida que crece el tamaño de los subconjuntos de la grilla, mejoran las métricas en el conjunto de entrenamiento, esto no necesariamente ocurre para las métricas en conjuntos de evaluación.

Validación cruzada sobre el modelo kNN para clasificación binaria

Para el modelo kNN entrenado por cada conjunto de datos analizado, medimos el desempeño para distintas cantidades de vecinos k. Los valores elegidos fueron 1, 3, 5, 10, 15, 20, 30 y 50. Este análisis fue realizado aplicando el método 5-Fold Cross Validation para cada valor de k.

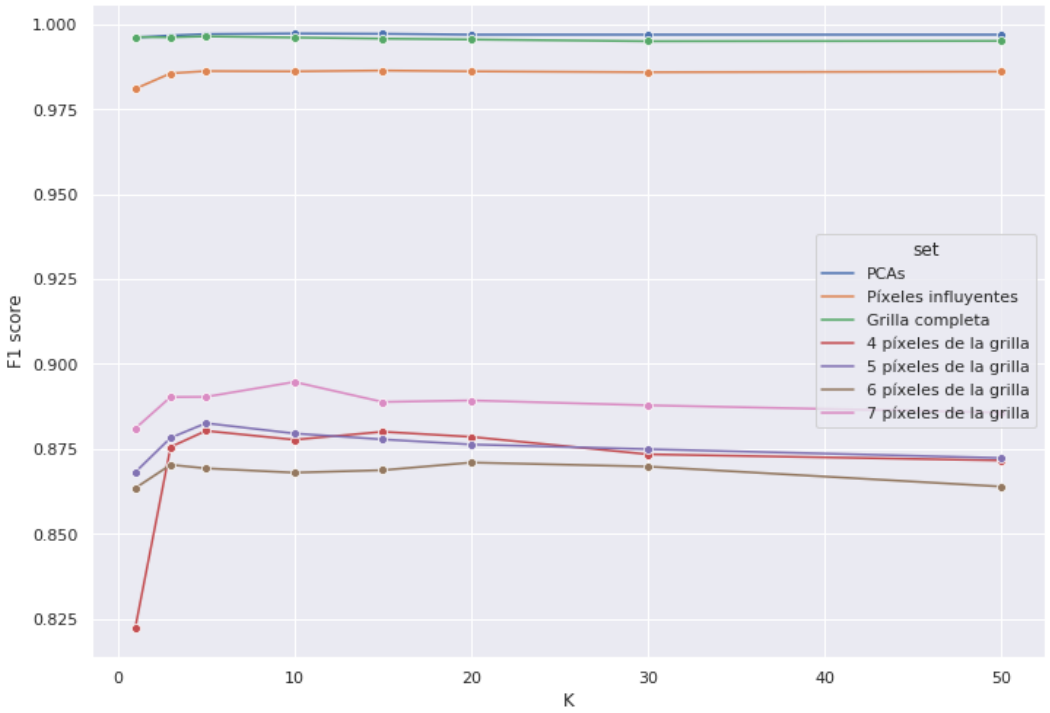

***Figura 8**: F1 Score del modelo kNN binario en función de la cantidad de vecinos (K), para cada uno de los conjuntos de datos utilizados, aplicando 5-Fold Cross Validation.*

La combinación que mejor rendimiento tuvo fue utilizar los valores del PCA con una cantidad de vecinos k=10. En cuanto a conjuntos de píxeles, se puede observar tanto aquí como en la ***Tabla 1*** que agregar píxeles aleatoriamente, como ocurre con los subconjuntos de la grilla, no implica que mejorará el rendimiento del modelo, puesto que no se ve una clara mejora de las métricas a medida que la cantidad de atributos aumenta. Sin embargo, teniendo un criterio definido, como ocurre con los píxeles que más influencia tienen en las componentes principales, observamos que el rendimiento fue considerablemente mayor aún siendo un conjunto de menor tamaño. La grilla completa ha sido el segundo conjunto de datos con mejor rendimiento, con un F1 Score máximo de 0,9964 para k=5. Utilizaremos este modelo para evaluar los nuevos datos provistos por la cátedra dado que, además de tener un rendimiento considerablemente alto, posee mayor explicabilidad que el modelo entrenado sobre las *PCAs*.

Clasificación completa utilizando árboles de decisión

Primero, considerando que existía un leve desbalance de algunas clases con respecto a otras, para optimizar la clasificación decidimos aplicar la técnica llamada submuestreo. Para las clases con mayor cantidad de ocurrencias que la mediana m=5936, decidimos seleccionar un conjunto de m elementos de cada clase, aleatoriamente. En total se han perdido 1365 de 60000 registros, de las clases ***1, 2, 3, 7*** y ***9.***

Para realizar la clasificación por árbol de decisión, seleccionamos nuevamente subconjuntos de atributos. Y, para cada uno de estos conjuntos de atributos, evaluamos distintas profundidades del árbol utilizando 10-Fold Cross Validation y calculando

únicamente exactitud como métrica de rendimiento, considerando ya balanceadas las clases. Utilizamos una mayor cantidad de divisiones de los datos en el Cross Validation, ya que hay más datos que en caso binario. El criterio utilizado para dividir los nodos fue la entropía, y los valores de las profundidades máximas han sido 3, 5, 7, 10, 15, 20 y 30.

Antes de elegir los conjuntos de atributos, contamos con la hipótesis de que el tiempo de ejecución al entrenar y predecir mediante el árbol de decisión dado el tamaño del conjunto de datos es menor al tiempo que toma el kNN. Para comprobarlo, primero medimos el tiempo que toman ambos modelos aplicándolos sobre la grilla presentada en la Figura 7, utilizando todos los dígitos. Utilizamos el kNN con k=5, y el árbol de decisión con profundidad máxima d=5. El tiempo tomado por el kNN fue 0,123 s, con una exactitud de 0,996 en un conjunto de evaluación. Y por el árbol de decisión 0,0153 s, con una exactitud de 0,990 también en un conjunto de evaluación, lo que representa un orden de magnitud menos con respecto al kNN en cuanto al tiempo. Por éste motivo, decidimos tomar más atributos para entrenar el árbol de decisión.

Primero, comenzamos calculando de nuevo cuatro componentes principales luego de aplicar el submuestreo, y los utilizamos para entrenar el modelo. Utilizamos la grilla presentada en la figura 7. Por último, también consideramos un nuevo subconjunto de atributos para entrenar el modelo: promediamos los valores de intensidad de cada píxel, y luego seleccionamos el conjunto de píxeles cuyo promedio se encontraba por encima del percentil 80. Este percentil se calculó sobre la distribución de los promedios de intensidades para cada píxel. Obtuvimos un conjunto de 157 píxeles.

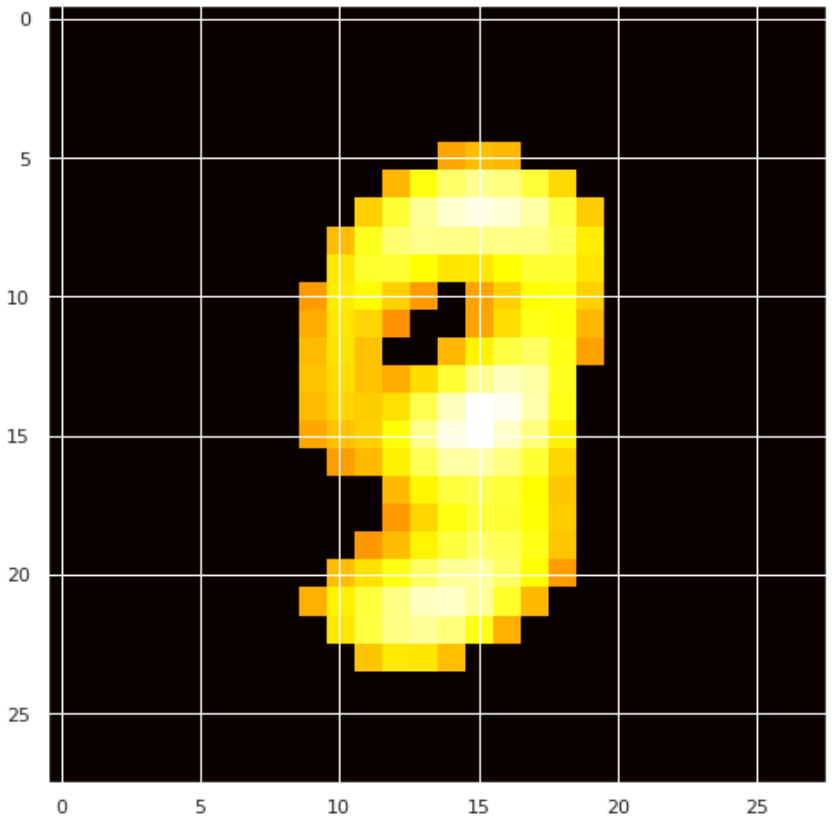

Figura 9*: Mapa de calor del promedio de intensidad de los píxeles con promedio de intensidad por encima del percentil 80*

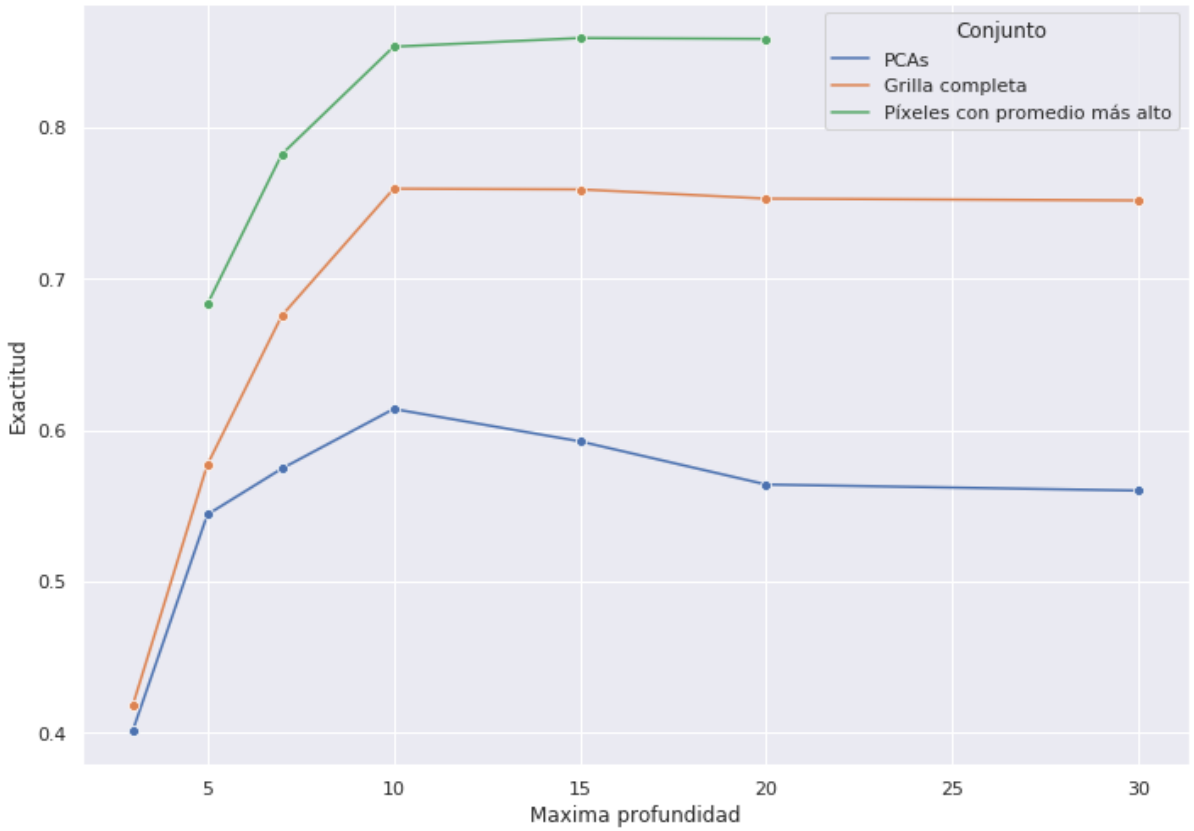

***Figura 10**: Exactitud en función de la Profundidad Máxima para cada uno de los conjuntos de entrenamiento utilizados*

Observamos que el modelo con mejor rendimiento fue utilizar los píxeles con intensidad promedio más alta, con una exactitud de 0,8588 y una máxima profundidad de 15. Notar que en los tres casos, la exactitud bajaba a partir de cierto punto.

Evaluación de modelos con mejor rendimiento

Finalmente, evaluamos sobre los conjuntos de test provistos por la cátedra los modelos que seleccionamos: el modelo de kNN con k=5 vecinos para el conjunto de datos binario entrenado sobre el conjunto de píxeles ubicados en la grilla de la ***Figura 7***, que es el modelo que mejor rendimiento tuvo de entre los modelos entrenados a partir de conjuntos de píxeles; y el Árbol de decisión de profundidad máxima d=15 para el conjunto de datos completo (con todas las clases) entrenado sobre el conjunto de píxeles con promedio de intensidad mayor al percentil 80.

Modelo	Exactitud del modelo en test	F1 Score del modelo en test	Exactitud del modelo en training	F1 Score del modelo en training
kNN	0,9844	0,9834	0,9976	0,9974
ÁdD	0,8731	-	0,9957	-

Tabla 2*: Resultados obtenidos con los modelos seleccionados, evaluados sobre los conjuntos de test provistos por la cátedra*

Se puede observar que el desempeño en el conjunto de test disminuye con respecto al desempeño sobre el conjunto de entrenamiento en ambos casos. Sin embargo, consideramos que los resultados obtenidos son satisfactorios, dado que son similares a lo que obtuvimos anteriormente durante el desarrollo de los modelos.

Conclusiones

En primer lugar, consideramos que los datos son adecuados para entrenar modelos de aprendizaje automático para clasificar dígitos manuscritos. Por un lado, porque al reducir la dimensionalidad con la técnica PCA a 2, pudimos observar que algunas clases se separan de manera apreciable. En segundo lugar, porque eligiendo píxeles específicos para ser utilizados como conjunto de entrenamiento del modelo, el rendimiento era considerablemente más alto que un clasificador entrenado a partir de un conjunto aleatorio de atributos.

Por otra parte, podemos decir que al entrenar modelos sobre el conjunto de datos binario (únicamente con las clases ***0*** y ***1***), el rendimiento de los modelos utilizados -más allá del modelo en sí mismo y sobre qué conjunto de atributos haya sido entrenado- tiende a ser superior que al utilizar el conjunto de datos completo (con todas las clases). Es decir, cuantas menos clases haya que clasificar, mejor rendimiento tendrá el modelo. Esto puede tener que ver también con las formas propias de cada dígito: al utilizar el conjunto de datos binario, se puede observar que el ***0*** y el ***1*** son considerablemente distintos en cuanto a forma, y por lo tanto el rendimiento para clasificarlos ha sido apreciablemente elevado; mientras que al considerar el conjunto de datos conformado por todas las clases, se puede visualizar la similitud que presentan algunos dígitos entre sí, como el ***3*** y el ***8***, o el ***8*** y el ***9***.

Además, consideramos que a partir de lo realizado en este trabajo no es correcto comparar el desempeño de los modelos de k-NN y de Árboles de Decisión según las métricas que fueron mencionadas a lo largo del informe, ya que los hemos utilizado en contextos distintos y para problemas distintos. Sin embargo, optamos por utilizar una métrica más que nos pudiera dar alguna forma de comparar los modelos, ya mencionada anteriormente: el *tiempo de ejecución.* Al entrenar ambos modelos con la grilla presentada en la figura 7 y medir los tiempos, fue posible observar que el modelo de Árbol de decisión tiene un rendimiento considerablemente superior en términos de tiempo (incluso sin que esto implique un sacrificio de la exactitud, ya que, como se observa en los resultados, ambos modelos obtienen exactitudes similares).

Por último, pudimos concluir que el rendimiento de los modelos seleccionados con sus respectivas combinaciones de atributos e hiperparámetros no variaba

considerablemente con respecto a los resultados que obtuvimos al elegirlos, lo que consideramos satisfactorio.

Fuentes empleadas

[1] MNIST - Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.