DEPARTAMENTO DE COMPUTACION

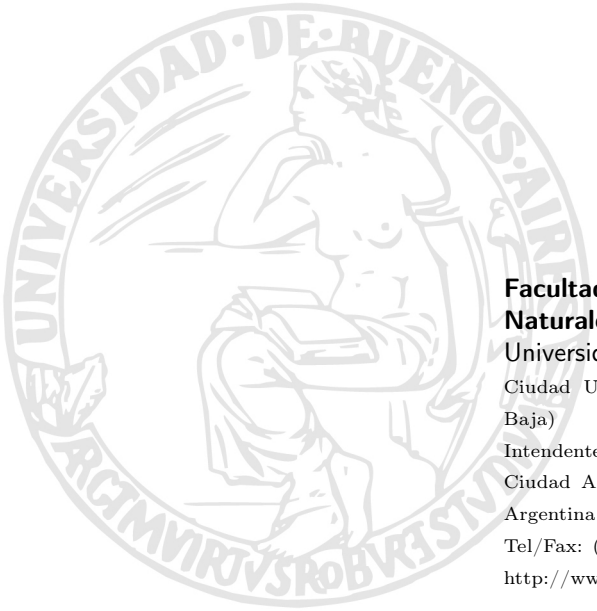
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Clasificación y validación cruzada

June 15, 2023 Laboratorio de Datos

Integrante	LU	Correo electrónico
Guibaudo, Camila	682/17	cami_sol_guibaudo@hotmail.com
Dembling, Ariel	408/92	arieldembling@gmail.com
Suarez, Antony	792/21	sebastsuar@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

1 Introducción

En este trabajo práctico, se tiene como objetivo el desarrollo y análisis de distintas técnicas de clasificación basadas en aprendizaje automático supervisado. Para ello se trabajará con un conjunto de datos llamado MNIST, que contiene información sobre imágenes de dígitos numéricos manuscritos. Se pretende crear modelos de clasificación capaces de, dada una imagen de un dígito escrito a mano, predecir con qué dígito se corresponde.
Como algoritmos de clasificación se utilizaron el algoritmo KNN (K-Nearest Neighbors) y el de árboles de clasificación, y, para cada uno de estos algoritmos se implementaron varios modelos diferentes, variando algunos parámetros e hiperparámetros. En el caso del algoritmo KNN se crearon varios modelos predictores, considerando varios subconjuntos de atributos diferentes; en el caso de los árboles de decisión, se crearon varios modelos con árboles de profundidades diferentes. Cada uno de estos modelos fue evaluado con conjuntos de datos de testing y se analizaron sus correspondientes performances, las cuales serán informadas más adelante en este informe.

1.1 Datasets a utilizar

Para el desarrollo de los modelos, se utilizará un dataset llamado *mnist_ desarrollo*; para el posterior testing, se utilizarán otros dos datasets llamados *mnist_ test* y *mnist_ test_ binario*, todos ellos provistos por la cátedra.

1.2 Análisis exploratorio de datos

En el análisis exploratorio de datos se realizó una descripción detallada sobre las características de los datos correspondientes al dataset *mnist_ desarrollo*. He aquí las principales observaciones resultantes:

- Filas y columnas: se observó que cada fila del dataset representa una imagen. En cuanto a las columnas se observó que la primera contiene las etiquetas correspondientes a las imágenes, y que el resto representa imágenes del dígito manuscrito, tal que cada una de esas columnas representa cada uno de los pixels de dichas imágenes.
  En la primera columna, columna de etiquetas, los valores son enteros del 0 al 9, correspondientes a los dígitos que representan. En el resto de las columnas, los valores posibles son enteros entre 0 y 255 y represetan la intensidad de luminosidad que tiene cada imagen en el pixel, correspondiente a esa columna, correspondiente a la profundidad, firmeza o densidad del trazo en cada punto. El trazo, entonces, se ve representado con valores en el intervalo [1,255], mientras que el fondo (*background*) está representado con el valor 0.

- Estructura del dataset:

  - Tamaño de la dataset: 60000 x 785

  - Cantidad de imágenes: 60000
  - Cantidad de clases: 10
  - Valores de las etiquetas: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Datos de los atributos: enteros entre 0 y 255
  - Cantidad total de pixels por imagen: 784
  - Las imágenes son cuadradas.
  - Cantidad de pixels por lado: 28

- Relevancia: no todos los pixels de las imágenes resultan igualmente relevantes para la identificación del dígito, ya que el trazo de los dígitos no se extiende a todos los pixels por igual. A fin de representar gráficamente la relación entre la cantidad de trazos del dataset y los pixels que ellos ocupan, se contó dicha cantidad para cada pixel. No se tuvo en cuenta la intensidad del trazo para esto: con que una muestra contenga un valor distinto de 0 en un pixel, ya esa muestra es de las que se cuenta para la evaluación de ese pixel. Al finalizar el cómputo, los valores obtenidos se reescalaron al intervalo [0,255], obteniendo así valores aptos para ser visualizados del mismo modo que las propias imágenes del dataset. El resultado puede verse en la Fig. 1. En la Fig. 2 puede observarse la imagen correspondiente al subset de los dígitos 0 y 1.

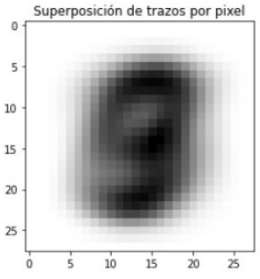


Figura 1: Superposición cuantizada de trazos por pixel, dataset completo

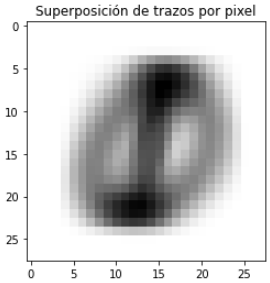


Figura 2: Superposición cuantizada de trazos por pixel, dataset binario

Se consideraron varios criterios posibles en base a ésto, de entre los cuales arribamos a unos que denominamos "trazo máximo" , "trazo medio" y trazo único". En trazo máximo se seleccionan pixels con las cantidades mayores de trazos, en trazo medio las cantidades intermedias y en trazo único las que corresponden a aproximadamente 1/n de la muestra, donde n es la cantidad de dígitos que se representan en la muestra (10 para el dataset original, 2 para el binario).

Se consideraron varias interpretaciones posibles para estos criterios. Por ejemplo, una interpretación de "trazo máximo" es que es más popular ese píxel en cuanto a que más imágenes del dataset tienen trazos que pasan por él, y por tanto contiene información de interés. Sin embargo, se objetó que los pixels que están en la mayoría de las imágenes posiblemente tengan menos información que los que están en una cantidad menor, pues en el límite se acercan a los pixels que aparecen siempre, los cuales tienen información nula. Del mismo modo, se hipotetizó que los pixels que aparezcan solo en 1/n de los casos (trazo único) corresponderían solo a uno de los dígitos, maximizando así su utilidad para clasificar imágenes. Sin embargo, a ello se objeta que no hay garantías de que todas esas muestras en las que aparezcan correspondan a un único dígito.

Véanse comentarios en el código para mayor detalle sobre la cuestión y la implementación en general.

Se calculó la relevancia de cada píxel según estos criterios.

- Se analizaron las cantidades de muestras correspondientes a cada una de las 10 etiquetas, en el dataset. Los resultados, para las etiquetas que van del 0 al 9, fueron respectivamente, 5923, 6742, 6958, 6121, 5842, 5421, 5918, 6265, 5851, 5949. Estas cantidades representan los siguientes porcentajes, respectivamente, 9.87%, 11.24%, 9.93%, 10.22%, 9.74%, 9.04%, 9.86%, 10.44%, 9.75%, 9.91% con respecto al número de muestras totales. Si bien idealmente el porcentaje de muestras para cada clase debería ser aproximadamente el mismo para evitar que los modelos den resultados sesgados, estas diferencias de porcentajes obtenidas nos parecieron lo suficientemente pequeñas como para ignorarlas.

2 Experimentos realizados

2.1 K-Nearest Neighbors

Se seleccionaron atributos en base a varios criterios, los ya mencionados por relevancia y otros al azar, con diversas cantidades de atributos.

Para cada criterio de selección de atributos, se corrió kNN y se generaron gráficos variando k.

No se aplicó reescalado a los atributos (pixels) pues todos están en la misma escala, la del intervalo [0,255].

Se observó que una mayor cantidad de atributos no determinaba necesariamente una mayor exactitud. Por el contrario, con solo un par de pixels ubicados en las zonas no compartidas por el 0 y el 1, ya se lograba una exactitud casi perfecta. Este efecto posiblemente sea facilitado en este dataset binario, ya que las zonas de superposición de los dígitos es mínima.

Se implementó cross-validation repitiendo kNN, obteniendo valores promedio para cada k y para cada selección de atributos.

Los resultados obtenidos con el criterio de trazo medio fueron indistinguibles de los del trazo único (razón por la cual se omite su gráfico en este informe). Ambos fueron superiores a los atributos tomados (de a 3) al azar, y más aun al del trazo máximo.

2.2 Árboles de decisión

2.2.1 Análisis de performance para distintos valores de profundidad

Se crearon varios árboles de decisión con diferentes profundidades y se analizaron sus performances utilizando cross-validation con k-folding. Esto se realizó de la siguiente forma:

1. Dado el conjunto de datos, este se particionó en 5 folds, cada uno de los cuales, contenía 4/5 de los datos destinados para entrenamiento (conjunto de datos al que llamaremos X_train) y 1/5 de los datos destinados para validación (conjunto de datos al que llamaremos X_valid). Los folds armados fueron estratificados, lo que significa que todos los folds preservaron el mismo porcentaje de muestras de cada clase.

2. Se creó un árbol de decisión de profundidad P, al que llamaremos $arbol_P$

3. Se entrenó a $arbol_P$ utilizando a X_train del fold 1.

4. Se utilizó el conjunto de validación X_valid del fold 1, para realizar predicciones del modelo $arbol_P$. Estas predicciones se guardaron en una variable Y_pred.

5. Comparando los valores de Y_pred con los de Y_valid, se calculó la accuracy como métrica de performance del $arbol_P$ para el fold 1.

6. Se repitieron los pasos del 3 al 5 por cada fold, obteniéndose así, cinco medidas de accuracy para el $arbol_P$, una por cada fold.

7. Se promediaron las cinco medidas de accuracy del $arbol_P$, obteniendo así una única medida de performance para el $arbol_P$, a la que llamaremos accuracy promedio.

8. Se repitieron los pasos del 2 al 7 para distintos valores de P. Esto dio como resultado varios árboles de decisión y sus correspondientes medidas de performance promedio obtenidas por cross-validation.

Una vez realizado esto, se graficaron los distintos valores de performance promedio obtenidos para cada valor de profundidad P, considerando valores de P entre 1 y 40. El gráfico obtenido puede visualizarse en la siguiente figura:

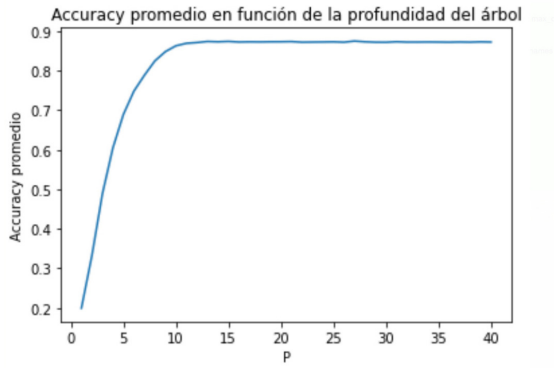


Figura 2: Accuracy promedio en función de la profundidad del árbol P

En este gráfico puede observarse que para valores de P mayores a 13, la accuracy promedio no varía mucho con el crecimiento de P. Entonces, podría decirse que un valor de P óptimo es 13, puesto que, el valor de accuracy obtenido para este valor de P es prácticamente el máximo posible y, además, con este valor de P el modelo no sería tan complejo como para otros valores de P mayores.

Nota: todos los árboles de decisión de este experimento fueron creados usando todos los datos y todos los atributos del dataset *mnist_ desarrollo*. Además, para todos ellos se usó el criterio de selección de atributos entropy.

2.2.2 Testing

kNN

A continuación se muestran los gráficos de kNN repetido 10 veces para los tres conjuntos de atributos seleccionados.

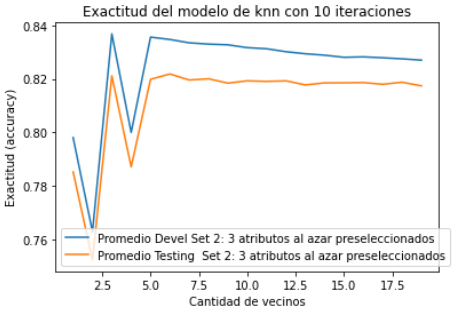


Figura 3: kNN repetido 10 veces, Set de atributos nro. 2

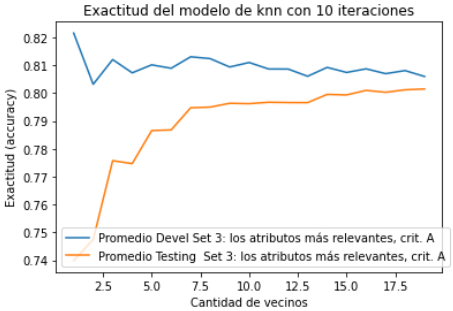


Figura 4: kNN repetido 10 veces, Set de atributos nro. 3

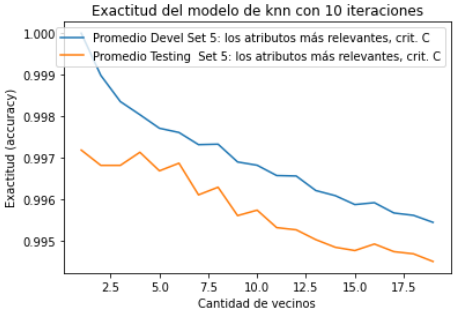


Figura 5: kNN repetido 10 veces, Set de atributos nro. 5

En la Tabla 1 se observan las medidas de performance observadas para kNN.

Árboles de decisión

Tomando el valor de P óptimo comentado en la subsección anterior, $P = 13$, se creó un árbol de decisión con esa profundidad. Como conjunto de entrenamiento se usaron todos los datos del dataset *mnist_desarrollo*. Como criterio para determinar qué atributo colocar en cada nodo, se usó entropy.

Con este modelo, se hicieron predicciones sobre los datos de los datasets *mnist_test* y *mnist_test_binario*. Comparando las clases predichas por el mod-

	Set 2: 3 pixels al azar	Set 3: trazo máximo	Set 5: trazo único/medio
Accuracy	0.81	0.79	0.99
Precisión	0.80	0.79	0.99
Cubrimiento	0.83	0.79	0.99

Table 1: Accuracy (exactitud) promedio máxima obtenida por cross-validation con kNN repetido 10 veces. Medido contra dataset "Test binario".

elo contra las clases reales expuestas en los datasets, se pudieron obtener las medidas de performance expuestas en la **Tabla 2**.

	Test	Test binario
Accuracy	0.87	0.99
Precisión	0.87	0.99
Cubrimiento	0.87	0.99

Table 2: Accuracy promedio en función de la profundidad del árbol P

Observamos en la **Tabla 1**, que las métricas de performance resultaron bastante bien para ambos tests, y, como era de esperar, para el test donde había 10 clases distintas, la performance fue un poco peor que para el test donde había solo 2 clases. Sin embargo, es importante recalcar que, el qué tan buena o mala sea la performance de un modelo, depende mucho del contexto de lo que se esté analizando. Por ejemplo, si este modelo se usara para interpretar escrituras de médicos sobre dosis de fármacos que deben tomar los pacientes, una exactitud del 87% como la obtenida, podría ser fatal.

3 Conclusiones

Se logró concretar el objetivo del trabajo que era crear modelos de predicción usando el algoritmo KNN y árboles de decisión y comparar sus performances al variar algunos parámetros e hiperparámetros de los modelos. El entrenamiento y validación de estos modelos se realizaron usando datasets destinados a desarrollo. Las validaciones se hicieron a través de cross-validation. Una vez realizado ese análisis de performances, se eligieron parámetros óptimos (en el caso del algoritmo KNN, el número de vecinos k, y en el caso de los árboles de decisión, la profundidad P) y se crearon modelos usando esos parámetros, que, posteriormente fueron testeados con datasets destinados a testing. Las performances de estos modelos fueron bastante buenas.