DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico 02

Verano 2025

Integrante	LU	Correo electrónico
Ballera, Alexander	668/24	alexballera@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón 0+∞)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

Resumen

En el presente informe se realiza un análisis de clasificación y selección de modelos, utilizando el dataset MNIST-C. La primera parte consta de un análisis exploratorio de los datos, en la que se analiza la cantidad de datos, cantidad y tipos de atributos, cantidad de clases de la variable de interés (dígitos) y otras características. La segunda parte consta de clasificación binaria y por último análisis multiclase en la que se realiza un análisis de selección de modelo mediante validación cruzada con k-folding.

Introducción

El análisis exploratorio de datos (AED) es un proceso que utiliza estadísticas y gráficos para conocer los datos y explorar su naturaleza. Es un paso previo necesario antes de realizar cualquier análisis estadístico o predictivo. Con el AED buscamos: explorar, investigar y aprender de los datos; identificar posibles errores; revelar la presencia de valores atípicos; comprobar la relación entre variables y realizar un análisis descriptivo de los datos.

Con esta información exploratoria y descriptiva de los datos buscamos clasificar y escoger modelos que nos permitan “entrenarlos”, validarlos que nos permitan predecir resultados esperados de acuerdo a los datos observados.

Análisis exploratorio

En primer lugar se descarga el archivo de datos **(mnist_c_fog_tp.csv)** provisto por el profesorado desde la sección de la materia en el campus virtual. Evaluo con df.shape y la misma posee 70.000 tuplas y 786 columnas.

```
In [7]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Columns: 786 entries, Unnamed: 0 to labels
dtypes: int64(786)
memory usage: 419.8 MB
```

Los datos proporcionados se caracterizan en 70.000 tuplas que van desde el índice 0 hasta 69.999, 786 columnas que van desde la columna ‘Unnamed: 0’ hasta la columna ‘labels’, todos del tipo entero.

Verifico si existe algún valor nulo o campo vacío ejecutando **df.isna().all().hasnans**, y el mismo retorna que el DataFrame no tiene valores nulos ni campos vacíos.

Columnas

```
In [6]: df.columns
Out[6]:
Index(['Unnamed: 0', '0', '1', '2', '3', '4', '5', '6', '7', '8',
       ...
       '775', '776', '777', '778', '779', '780', '781', '782', '783',
       'labels'],
      dtype='object', length=786)
```

Observo que la columna **'Unnamed: 0'** es una variable incremental en 1 igual al index, supondría que era el index de los datos originales. Además, observo que la última columna es **'labels',** la cual es diferente a las otras columnas que van desde 0 hasta 783.

Preparación de los datos

X = df.copy() # Copio el df a la variable X para mantener el df original

X.pop('Unnamed: 0') # Elimino la columna 'Unnamed: 0'

y = X.pop('labels') # Número representado en la imagen, escojo la columna 'labels' como la variable 'y'

```
In [12]: X.shape
Out[12]: (70000, 784)

In [13]: X.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Columns: 784 entries, 0 to 783
dtypes: int64(784)
memory usage: 418.7 MB
```

De esta manera, los datos para el análisis quedarían en 70.000 tuplas y 784 columnas = (28 x 28), los cuales contienen los píxels de la imagen, habiéndose registrado éstos de izquierda a derecha y de arriba abajo.

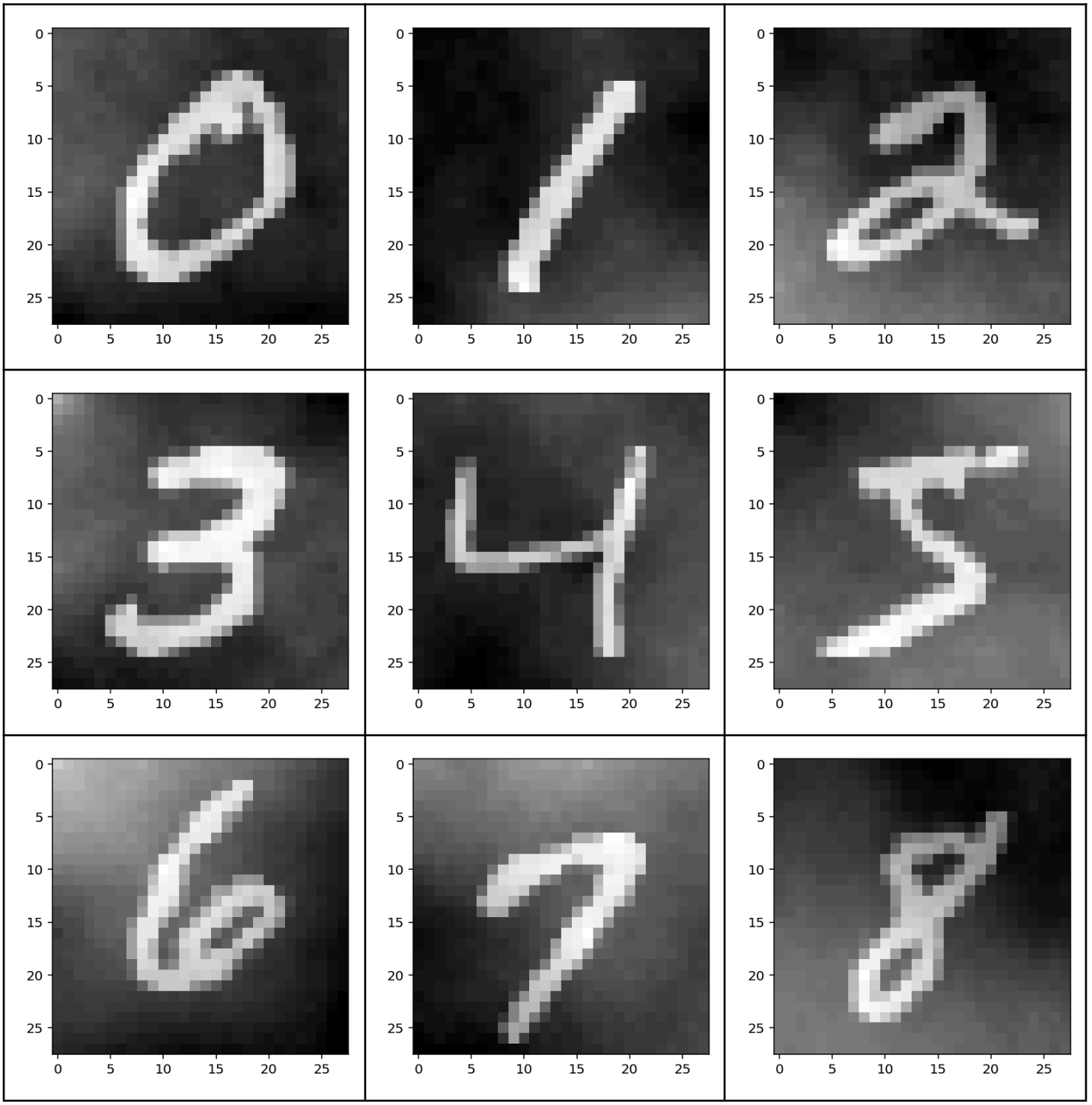
Y la variable “y” contiene la etiqueta de cada muestra (el número representado en la imagen), desde 0 hasta 9.

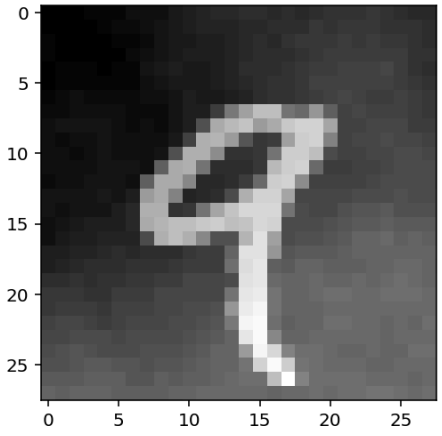
Algunos resultados de las imágenes

Ejecuto el siguiente comando para valores i desde 0 hasta 19, para verificar los primeros 20 valores y observar los resultados preliminares.

plt.imshow(X.iloc[i].values.reshape(28, 28), cmap="binary_r");

Algunos resultados:





a. Atributos

Tal como describí anteriormente en la sección columnas, el dataset tiene 786 atributos o columnas, se divide en 3 grupos, la primera columna denominada **'Unnamed: 0'**, desde las columnas 1 hasta la columna 785 son los pixeles (784 = 28x28) y la columna 786 etiquetada como **'labels'** contiene la etiqueta de cada muestra (el número representado en la imagen).

De los atributos se puede descartar la columna **Unnamed: 0**, ya que es el index del dataset original, las siguientes columnas no se pueden descartar ya que representan los pixeles de las imágenes y la columna **'labels'** se puede separar del dataset para comprobar algún tipo de predicción para validar, es descartable para mostrar la imagen ya que no es parte de los pixeles, pero se separa solo para validar y comprobar algún modelo predictivo.

b. Números parecidos

Si, los números parecidos que puede presentar confusión son uno (1) y siete (7), dos (2) y cinco (5), tres (3) y ocho (8), cuatro (4) y nueve (9).

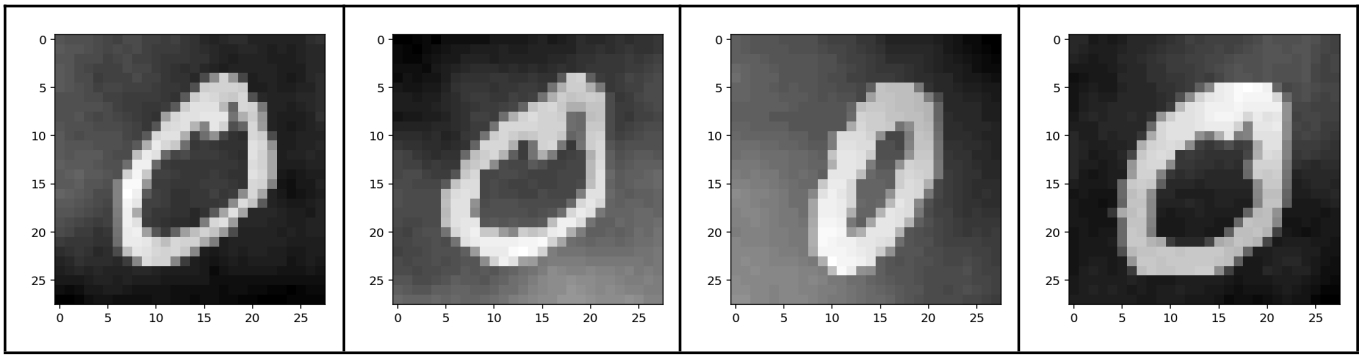
c. Digitos similares

Además de lo visible, de los números parecidos, evalúo los datos correspondiente a los pixeles y según la variación de cada línea o de cada imágen de cada número se puede observar qué tan similar o diferente es cada dígito, de cada imagen de cada línea del dataset. (Mean = promedio simple, SD = desviación standard, CV evalúo el coeficiente de variación que mide la proporción o variación de la SD respecto de la media de cada dígito)

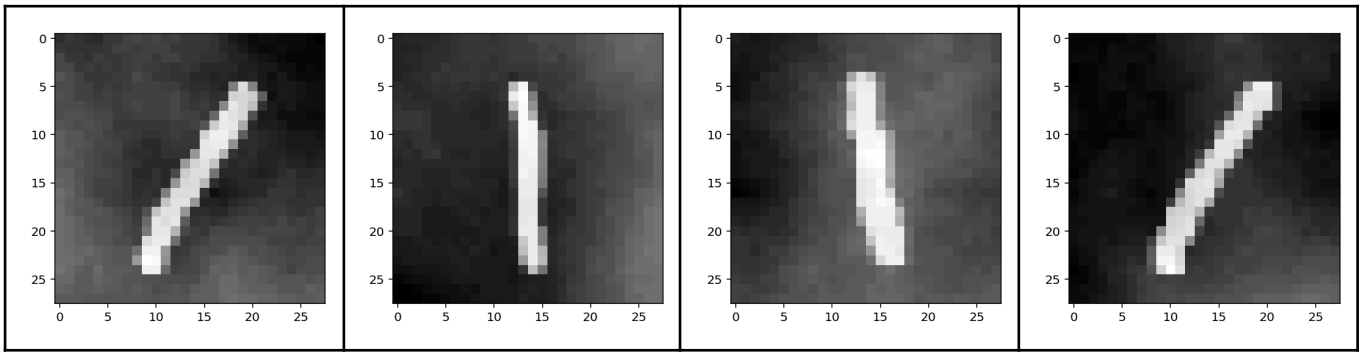
	0	1	2	3	4	5	6	7	8	9
Mean	106.11	99.71	104.70	104.12	102.90	103.27	104.53	102.78	104.96	103.87
SD	41.32	40.02	41.51	41.29	41.04	41.35	41.54	40.97	41.26	41.20
CV	0.38	0.40	0.39	0.39	0.39	0.40	0.39	0.39	0.39	0.39

Según el Coeficiente de Variación, todos los dígitos muestran valores superiores al 30%, lo que indica que son valores pocos representativos, es decir que cada tupla de cada dígito presenta variaciones o diferencias entre ellos. El set de conjuntos del dígito cero '0' es el que presenta más similitud entre ellos y los conjuntos uno (1) y cinco (5) son quienes presentan mayores variaciones.

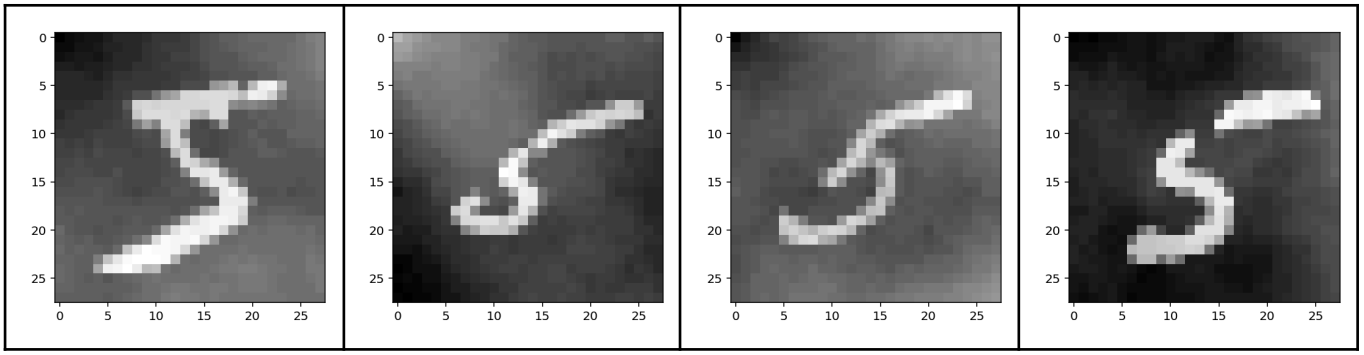
Algunos ceros (0):



Algunos unos (1):



Algunos cincos(5):



d. Exploración de datos

La principal diferencia es que cada columna representa un pixel de la imagen, con lo que todas las columnas están correlacionadas para mostrar un resultado final que es la imagen de un dígito, a diferencia de otro dataset como el de Titanic que cada atributo representa un atributo de la realidad y/o de la entidad, y de manera independiente puede explicar alguna cualidad de la entidad.

Por lo tanto, la manera de realizar un análisis exploratorio entre ambos datasets es diferente, pero es posible por el objetivo que busca cada set y cada análisis.

Clasificación binaria

a. Subconjuntos imágenes correspondiente a dígitos cero (0) y uno (1)

	labels = 0	label = 1
X.info()	Index: 6903 entries, 1 to 69993 Columns: 784 entries, 0 to 783 dtypes: int64(784)	Index: 7877 entries, 3 to 69994 Columns: 784 entries, 0 to 783 dtypes: int64(784)
X.mean().mean()	106.11	99.71
X.std().mean()	41.32	40.02
CV	0.38	0.40
Vacíos o Nulls	False	False

Observo que en ambos subconjuntos poseen los datos de sus campos completos y no nulos, además el coeficiente de variación (CV) de ambos está por encima del 30%, lo que indica que estadísticamente no son homogéneos, ya que presentan gran dispersión a la media. Estos valores de CV indica que las imágenes se van a presentar con bastante variación en cada tupla que se grafique, la imagen cero (0) al presentar un CV menor va a presentar menos diferencias en sus imágenes, tal como se muestra en las gráficas anteriores de la sección Digitos similares

b. Train & test subconjuntos

Realizo split de los subconjuntos, por un lado las etiquetas (‘labels’) en el subconjunto “y” y el resto de las columnas (784) corresponderían al subconjunto X.

Luego produzco los subconjuntos X_train, X_test, y_train, y_test, con el comando train_test_split(X, y, test_size = 0.3). Se dividen los datos de la siguiente forma: 30% al subconjunto test y 70% al subconjunto de entrenamiento. A continuación muestro la información de los diferentes subconjuntos:

_.info()	**labels = 0**	**label = 1**
X_train	Index: 4832 entries, 47572 to 44266 Columns: 784 entries, 0 to 783 dtypes: int64(784)	Index: 5513 entries, 47063 to 58253 Columns: 784 entries, 0 to 783 dtypes: int64(784)
X_test	Index: 2071 entries, 52804 to 40780 Columns: 784 entries, 0 to 783 dtypes: int64(784)	Index: 2364 entries, 53199 to 8156 Columns: 784 entries, 0 to 783 dtypes: int64(784)
y_train	<class 'pandas.core.series.Series'> Index: 4832 entries, 47572 to 44266 Series name: labels Non-Null Count Dtype -------------- ----- 4832 non-null int64 dtypes: int64(1)	<class 'pandas.core.series.Series'> Index: 5513 entries, 47063 to 58253 Series name: labels Non-Null Count Dtype -------------- ----- 5513 non-null int64 dtypes: int64(1)
y_test	<class 'pandas.core.series.Series'> Index: 2071 entries, 52804 to 40780 Series name: labels Non-Null Count Dtype -------------- ----- 2071 non-null int64 dtypes: int64(1)	<class 'pandas.core.series.Series'> Index: 2364 entries, 53199 to 8156 Series name: labels Non-Null Count Dtype -------------- ----- 2364 non-null int64 dtypes: int64(1)

c. Analizar modelo de KNN, con diferentes atributos: 3 y/o diferentes a 3

Luego produzco los subconjuntos X_train, X_test, y_train, y_test, con el comando train_test_split(X, y, test_size = 0.3). Se dividen los datos de la siguiente forma: 30% al subconjunto test y 70% al subconjunto de entrenamiento. A continuación muestro la información de los diferentes subconjuntos:

	labels = 0	**label = 1**
3 atributos _.info()	Index: 6903 entries, 1 to 69993 Data columns (total 3 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 0 6903 non-null int64 1 1 6903 non-null int64 2 2 6903 non-null int64 dtypes: int64(3)	Index: 7877 entries, 3 to 69994 Data columns (total 3 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 0 7877 non-null int64 1 1 7877 non-null int64 2 2 7877 non-null int64 dtypes: int64(3)
accuracy 3 attr	1.0	1.0
accuracy 10 attr	1.0	1.0
accuracy 20 attr	1.0	1.0

Realizo análisis variando la cantidad de atributos de 3, 10 y 20 atributos, tanto para el conjunto de imágenes cero (0) y uno (1) arrojando los valores de **accuracy** que se indican en la tabla anterior. Según estos valores, el modelo para los diferentes escenarios presenta **ceros falsos positivos** y **ceros falsos negativos**.

d. Analizar modelo de KNN, diferentes atributos y diferentes k

Realizo análisis variando la cantidad de atributos (3, 10 y 20) para construir los subconjuntos X_train, X_test, y_train, y_test respectivos, y luego para cada subconjunto creo modelos KNeighborsClassifier variando k = 3, 5, 10, los resultados de accuracy de cada subconjunto y modelo se muestran a continuación:

k = 3	**labels = 0**	**label = 1**
accuracy 3 attr	1.0	1.0
accuracy 10 attr	1.0	1.0
accuracy 20 attr	1.0	1.0

k = 5	**labels = 0**	**label = 1**
accuracy 3 attr	1.0	1.0
accuracy 10 attr	1.0	1.0
accuracy 20 attr	1.0	1.0

k = 10	**labels = 0**	**label = 1**
accuracy 3 attr	1.0	1.0
accuracy 10 attr	1.0	1.0
accuracy 20 attr	1.0	1.0

Realizo análisis variando la cantidad de atributos de 3, 10 y 20 atributos, tanto para el conjunto de imágenes cero (0) y uno (1), además varío el valor de k = 3, 5 y 10, arrojando los valores de **accuracy** que se indican en las tablas anteriores. Según estos valores, el modelo para los diferentes escenarios presenta **ceros falsos positivos** y **ceros falsos negativos**.