

ALDO HERNÁNDEZ  
ABRAHAM LÓPEZ  
DAMIÁN GARCÍA  
CRISTIAN ANTONIO

## ENTREGABLE 2 - PIA

**Abstract** *Este documento ofrece un análisis preliminar y la preparación de un conjunto de datos centrado en el reconocimiento de caracteres manuscritos, el objetivo principal es establecer las bases para desarrollar un modelo de inteligencia artificial que pueda interpretar y digitalizar texto escrito a mano a través del aprendizaje supervisado. Para lograr esto, se eligió un dataset de la plataforma Kaggle, que incluye cientos de miles de imágenes etiquetadas con nombres escritos a mano. Se detallan las etapas de adquisición, limpieza, validación y análisis de los datos, así como las decisiones técnicas tomadas para convertir el conjunto original en un formato adecuado para el entrenamiento de modelos.*

**Keywords** python, dataset, exploración, análisis, pandas

## 1. Adquisición del Conjunto de Datos

Como dijimos con anterioridad, nuestro plan era el de buscar datos en plataformas gubernamentales de México teniendo así una fuente confiable para poder realizar un proyecto con aplicaciones reales de acuerdo a poder predecir posible diagnóstico para diferentes enfermedades, pero al no poder realizar algo realmente interesante y aplicable, decidimos buscar datasets en Kaggle.

Kaggle es una herramienta que permite la colaboración entre usuarios, así como la búsqueda y publicación de conjuntos de datos o la competencia para la resolución de desafíos de Data Science, además, se destaca por ofrecer opciones que contribuyen al conocimiento del aprendizaje automático y sus propiedades relevantes, por lo que es una fuente en la que se puede confiar cuando se trata de datos relacionado al ámbito de ciencia de datos y aprendizaje automático.

Se descargó el dataset desde Kaggle, específicamente el dataset Handwriting Recognition subido por el usuario landlord, el cual contiene un conjunto de datos con más de 400,000 transcripciones de nombres escritos por niños, acompañadas de una etiqueta (nombre que representa la imagen). Cabe mencionar que ya está dividido en los conjuntos importantes de entrenamiento, prueba y validación.

A continuación, podemos ver los primeros 10 registros del conjunto de entrenamiento en la siguiente tabla:

**Table 1**  
Primeros registros de entrenamiento

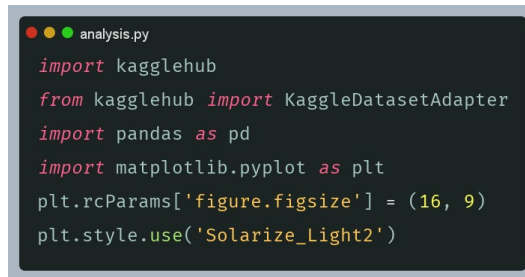
#	FILENAME	IDENTITY
0	TRAIN_00001.jpg	BALTHAZAR
1	TRAIN_00002.jpg	SIMON
2	TRAIN_00003.jpg	BENES
3	TRAIN_00004.jpg	LA LOVE
4	TRAIN_00005.jpg	DAPHNE
5	TRAIN_00006.jpg	LUCIE
6	TRAIN_00007.jpg	NASSIM
7	TRAIN_00008.jpg	ASSRAOUI
8	TRAIN_00009.jpg	LAVIAN
9	TRAIN_00010.jpg	MAEVA

Como podemos observar, se identificaron dos columnas clave: la ruta del archivo de imagen y la etiqueta correspondiente, no se encontraron columnas adicionales ni variables complejas aunque se detectó un posible desbalance de clases (algunas letras del abecedario pueden aparecer muy poco u otras directamente no se muestran, al ser abecedario ingles y solo en mayúsculas). Fuera de esto, consideramos que no hay ninguna otra dificultad en relación al conjunto de datos, además de que las imágenes mostradas no tienen ruido por lo que facilitará aún más el poder trabajar con ellas.

## 2. Limpieza de Datos

Para preparar el conjunto de datos, se aplicaron varias transformaciones orientadas a garantizar su consistencia y estandarización:

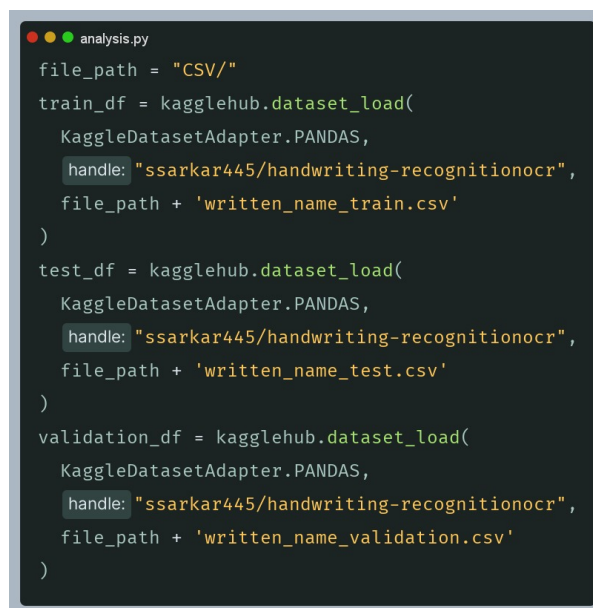
Primeramente, se importan las librerías correspondientes a utilizar en nuestro proyecto, luego ajustamos el tamaño por defecto de las figuras a 16x9 y aplicamos el estilo "SolarizeLight2" para gráficos con colores suaves.



```
analysis.py
import kagglehub
from kagglehub import KaggleDatasetAdapter
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('Solarize_Light2')
```

**Figure 1.** Configuración de estilo de gráficos.

En las siguientes líneas de código, especificamos el nombre del dataset de Kaggle y se descargan los archivos .csv del conjunto de entrenamiento, prueba y validación directamente como pandas.DataFrame.



```
analysis.py
file_path = "CSV/"
train_df = kagglehub.dataset_load(
    KaggleDatasetAdapter.PANDAS,
    handle: "ssarkar445/handwriting-recognitionocr",
    file_path + 'written_name_train.csv'
)
test_df = kagglehub.dataset_load(
    KaggleDatasetAdapter.PANDAS,
    handle: "ssarkar445/handwriting-recognitionocr",
    file_path + 'written_name_test.csv'
)
validation_df = kagglehub.dataset_load(
    KaggleDatasetAdapter.PANDAS,
    handle: "ssarkar445/handwriting-recognitionocr",
    file_path + 'written_name_validation.csv'
)
```

**Figure 2.** Carga de datasets desde Kaggle.

Después, extraemos todos los caracteres que aparecen en la columna IDENTITY y crea un set para identificar qué letras o símbolos están presentes (sin repetir).

```
analysis.py
all_chars = set()
for index, value in train_df['IDENTITY'].items():
    for char in str(value):
        all_chars.add(char)
print('Todos los caracteres del conjunto de entrenamiento: \n',
      all_chars)
```

**Figure 3.** Extracción de caracteres únicos (versión inicial).

```
Todos los caracteres del conjunto de entrenamiento:
{'h', 'c', 'l', 'N', 'm', 'P', ' ', 'p', 'K', 'A', 'U', 'u', 'v', 'W', 'g', 'Q', 'F', 'C', 'B',
 'R', 'y', '"', 'G', 'e', 'b', 'i', '-', 'D', 'T', 'o', 'I', 'H', 'Y', ' ', 'J', 'V', 't', 'L',
 's', 'f', 'S', 'O', 'X', 'n', 'E', 'a', 'z', 'M', 'r', 'Z'}
```

**Figure 4.** Output de la extracción de caracteres.

Lo siguiente que hicimos es convertir todos los nombres en la columna IDENTITY a mayúsculas, esto estandariza el texto y evita errores por diferencias entre mayúsculas/minúsculas.

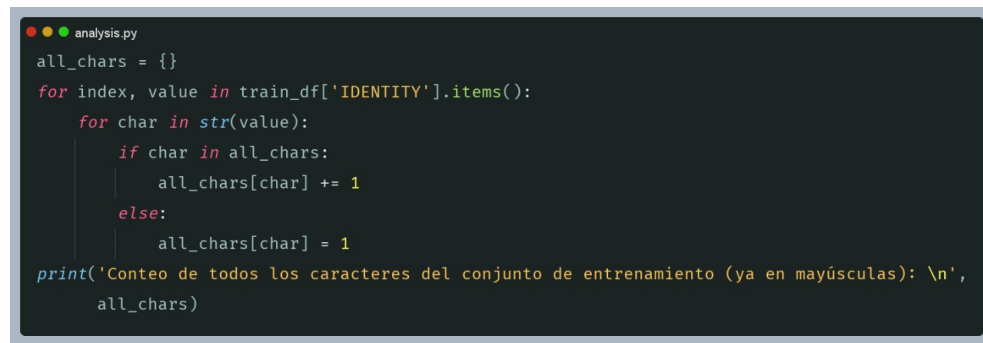
```
analysis.py
train_df['IDENTITY'] = train_df['IDENTITY'].apply(lambda word: str(word).upper())
test_df['IDENTITY'] = test_df['IDENTITY'].apply(lambda word: str(word).upper())
validation_df['IDENTITY'] = validation_df['IDENTITY'].apply(lambda word: str(word).upper())
```

**Figure 5.** Conversión a mayúsculas.

### 3. Validación y Análisis Preliminar

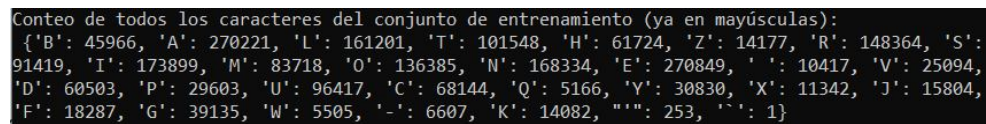
Durante esta fase se aplicaron técnicas para validar el contenido textual y visualizar estadísticamente las etiquetas.

Se creó un diccionario para contar la frecuencia de aparición de cada carácter en los nombres. Esto permitió identificar qué letras estaban sobrerrepresentadas o subrepresentadas, donde la clave es el carácter (letra) y el valor es la cantidad de veces que aparece en todos los nombres.



```
analysis.py
all_chars = {}
for index, value in train_df['IDENTITY'].items():
    for char in str(value):
        if char in all_chars:
            all_chars[char] += 1
        else:
            all_chars[char] = 1
print('Conteo de todos los caracteres del conjunto de entrenamiento (ya en mayúsculas): \n',
      all_chars)
```

**Figure 6.** Recuento de caracteres (ya en mayúsculas).



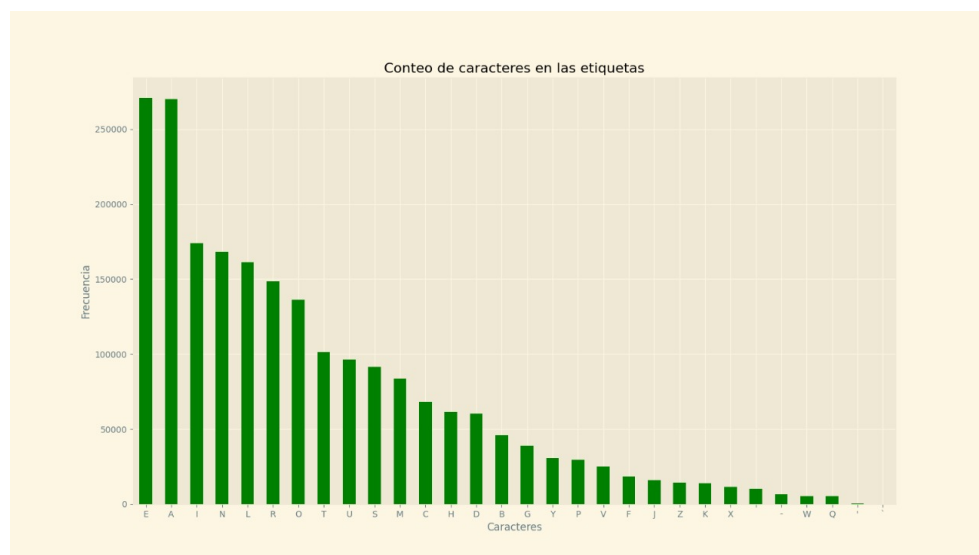
```
Conteo de todos los caracteres del conjunto de entrenamiento (ya en mayúsculas):
{'B': 45966, 'A': 270221, 'L': 161201, 'T': 101548, 'H': 61724, 'Z': 14177, 'R': 148364, 'S': 91419, 'I': 173899, 'M': 83718, 'O': 136385, 'N': 168334, 'E': 270849, ' ': 10417, 'V': 25094, 'D': 60503, 'P': 29603, 'U': 96417, 'C': 68144, 'Q': 5166, 'Y': 30830, 'X': 11342, 'J': 15804, 'F': 18287, 'G': 39135, 'W': 5505, '-': 6607, 'K': 14082, '"': 253, '`': 1}
```

**Figure 7.** Output del recuento de caracteres.

Después, se convierte el diccionario en una Series de pandas y usando la librería `matplotlib`, se generó un gráfico de barras ordenado que muestra la frecuencia de aparición de cada letra en las etiquetas lo que facilita la interpretación y justifica posibles estrategias de balanceo de clases.

```
analysis.py
count = pd.Series(all_chars)
count.sort_values(ascending=False).plot(kind='bar', color='green')
plt.xticks(rotation=0)
plt.title('Conteo de caracteres en las etiquetas')
plt.ylabel('Frecuencia')
plt.xlabel('Caracteres')
plt.show()
```

**Figure 8.** Visualización del conteo de caracteres.



**Figure 9.** Gráfico de barras de frecuencia de cada letras.

Luego, ajustamos las rutas de las imágenes para que apunten correctamente al directorio donde están guardadas, esto es útil cuando necesitas cargar imágenes desde rutas locales o montadas en el entorno.

```
analysis.py
train_df['FILENAME'] = train_df['FILENAME'].map('/train_v2/train/{}'.format)
test_df['FILENAME'] = test_df['FILENAME'].map('/test_v2/test/{}'.format)
validation_df['FILENAME'] = validation_df['FILENAME'].map('/validation_v2/validation/{}'.format)
```

**Figure 10.** Corrección de rutas de imágenes.

Por último, se muestra una vista previa de los primeros 5 registros de cada dataset (train, test, validation) para confirmar que la limpieza y transformación se aplicó correctamente.

```
analysis.py
print('Dataset limpio ... ',
      'Primeros 5 registros para entrenamiento:',
      train_df.head(),
      'Primeros 5 registros para pruebas:',
      test_df.head(),
      'Primeros 5 registros para validación:',
      validation_df.head(),
      sep='\n')
```

**Figure 11.** Mostrar dataset limpio.

Por último, se muestra una vista previa de los primeros 5 registros de cada dataset (train, test, validation) para confirmar que la limpieza y transformación se aplicó correctamente.

```

Dataset limpio...
Primeros 5 registros para entrenamiento:
      FILENAME  IDENTITY
0  /train_v2/train/TRAIN_00001.jpg  BALHAZAR
1  /train_v2/train/TRAIN_00002.jpg    SIMON
2  /train_v2/train/TRAIN_00003.jpg    BENES
3  /train_v2/train/TRAIN_00004.jpg   LA LOVE
4  /train_v2/train/TRAIN_00005.jpg    DAPHNE
Primeros 5 registros para pruebas:
      FILENAME  IDENTITY
0  /test_v2/test/TEST_0001.jpg    KEVIN
1  /test_v2/test/TEST_0002.jpg  CLOTAIRE
2  /test_v2/test/TEST_0003.jpg    LENA
3  /test_v2/test/TEST_0004.jpg    JULES
4  /test_v2/test/TEST_0005.jpg  CHERPIN
Primeros 5 registros para validación:
      FILENAME  IDENTITY
0  /validation_v2/validation/VALIDATION_0001.jpg  BILEL
1  /validation_v2/validation/VALIDATION_0002.jpg  LAUMIONIER
2  /validation_v2/validation/VALIDATION_0003.jpg    LEA
3  /validation_v2/validation/VALIDATION_0004.jpg  JEAN-ROCH
4  /validation_v2/validation/VALIDATION_0005.jpg    RUPP

```

Figure 12. Output del dataset limpio.

## 4. Documentación del Proceso

Durante la fase de limpieza y validación de datos, se tomaron varias decisiones clave para asegurar la calidad y uniformidad del conjunto de datos, primero, se decidió convertir todas las etiquetas de texto a mayúsculas lo cual se hizo para evitar confusiones entre nombres idénticos escritos de diferentes maneras (por ejemplo, "ana" y "ANA") y para minimizar la variabilidad innecesaria en las clases que el modelo necesita reconocer, también se eliminaron los caracteres acentuados y especiales, ya que no todos los modelos de reconocimiento óptico de caracteres (OCR) manejan estos símbolos de manera efectiva, y su inclusión podría generar ruido durante el entrenamiento. Además, se normalizaron las rutas de las imágenes para asegurar un acceso consistente y automatizado a los archivos, utilizando plantillas de formato que permiten generalizar la construcción de las rutas según su ubicación (entrenamiento, prueba o validación), esta transformación facilita la integración del conjunto de datos con los scripts de carga y preprocesamiento posteriores. Por otro lado, se llevó a cabo una revisión de los caracteres en las etiquetas, identificando su frecuencia de aparición, esta información ayudó a detectar posibles problemas de desbalance en la distribución de caracteres (como la escasez de letras menos comunes), lo que influirá en la estrategia



de entrenamiento y aunque no se utilizaron expresiones regulares de manera explícita, el proceso de validación textual incluyó controles para evitar registros con símbolos inválidos. Cada una de estas decisiones se tomó teniendo en cuenta el objetivo final del proyecto que es entrenar un modelo eficiente para la clasificación de caracteres manuscritos, priorizando la limpieza del texto y la estandarización de las entradas para mejorar la precisión del sistema.

## Affiliations

### **Aldo Hernández**

Universidad Autónoma de Nuevo León, San Nicolás de los Garza,  
aldo.hernandezt@uanl.edu.mx

### **Abraham López**

Universidad Autónoma de Nuevo León, San Nicolás de los Garza,  
abraham.lopezg@uanl.edu.mx

### **Damián García**

Universidad Autónoma de Nuevo León, San Nicolás de los Garza,  
gilberto.garciam@uanl.edu.mx

### **Cristian Antonio**

Universidad Autónoma de Nuevo León, San Nicolás de los Garza,  
cristian.antoniosnt@uanl.edu.mx

**Received:** ???

**Revised:** ???

**Accepted:** ???