

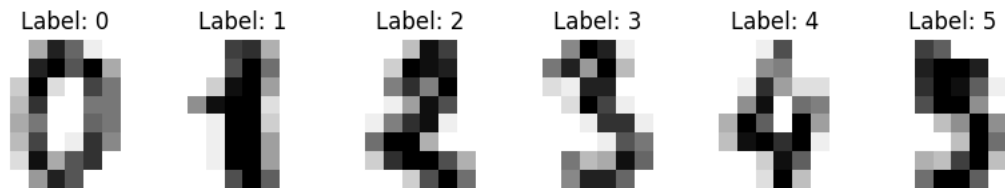
Load dataset

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits

# Cargar el dataset
digits = load_digits()

# Mostrar las primeras 4 imágenes junto con sus etiquetas
fig, axes = plt.subplots(1, 6, figsize=(10, 3))
for ax, image, label in zip(axes, digits.images, digits.target):
    ax.set_axis_off() # Desactiva los ejes
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest') # Muestra la imagen
    ax.set_title(f'Label: {label}') # Título con la etiqueta

plt.show()
```



Entender el dataset

```
# Cargar el dataset de dígitos de sklearn
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt

# Cargar el dataset
digits = load_digits()

"""
Entendiendo el dataset:
- El dataset contiene 1797 imágenes de dígitos escritos a mano (del 0 al 9).
- Cada imagen tiene una resolución de 8x8 píxeles (64 píxeles en total) y está representada en escala de grises.
  Los valores de los píxeles van de 0 (negro) a 16 (blanco brillante).
- Los datos de las imágenes están almacenados en `digits.data` como una matriz aplanada de 64 valores por imagen.
- `digits.images` contiene las imágenes originales en una matriz 8x8.
- `digits.target` contiene las etiquetas de cada imagen, indicando el número que representa (de 0 a 9).
"""

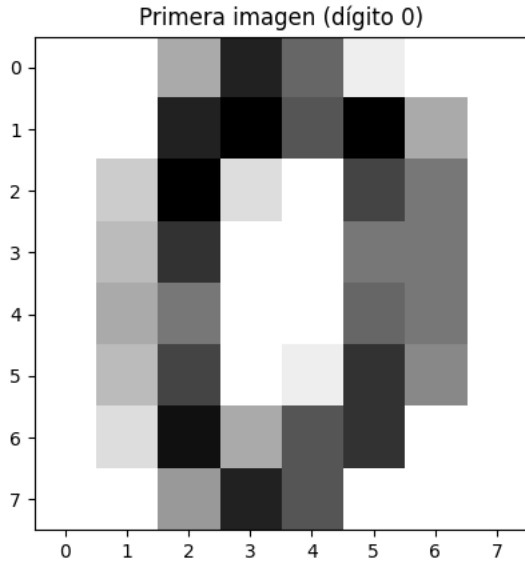
# Ver la primera imagen
print(f"Etiqueta de la primera imagen: {digits.target[0]}")
print(f"Datos de la primera imagen (aplanada): {digits.data[0]}")

# Mostrar la imagen original de 8x8 píxeles
plt.imshow(digits.images[0], cmap='gray_r')
plt.title(f"Primera imagen (dígito {digits.target[0]})")
plt.show()
```

```

Etiqueta de la primera imagen: 0
Datos de la primera imagen (aplanada): [ 0.  0.  5. 13.  9.  1.  0.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.
 15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.
  0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.
  0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]

```



Plotear algunos ejemplos (digits)

```

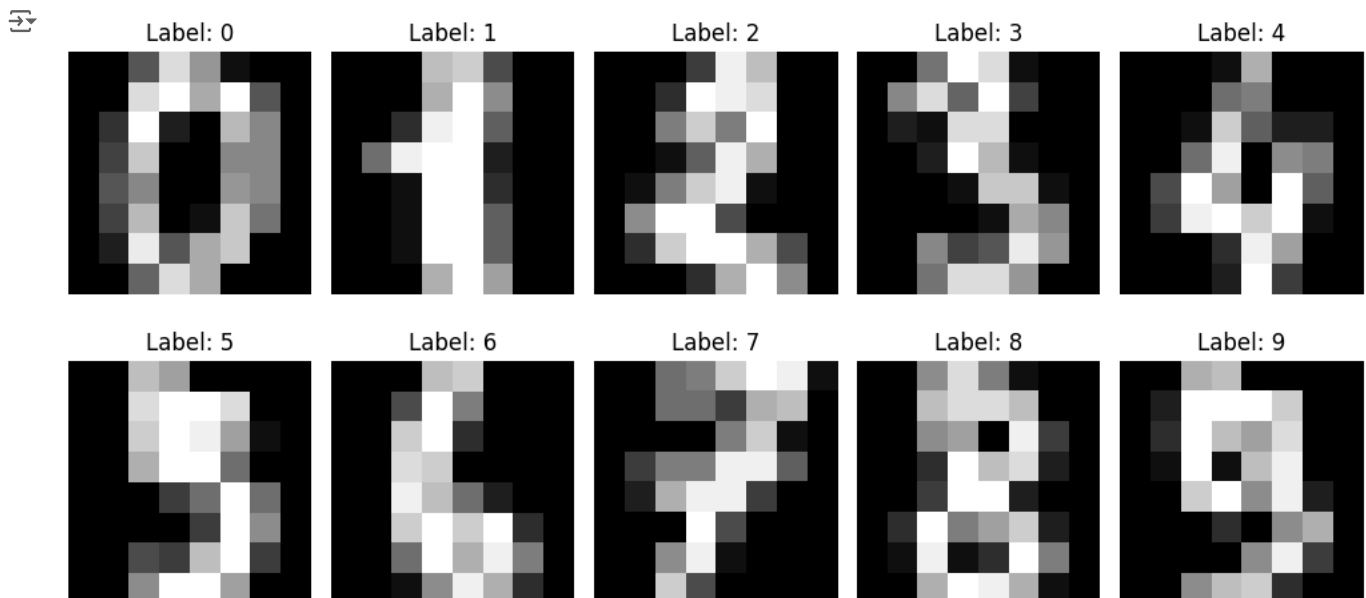
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits

# Cargar el dataset
digits = load_digits()

# Configuración para mostrar las primeras 10 imágenes
fig, axes = plt.subplots(2, 5, figsize=(10, 5))
for i, ax in enumerate(axes.ravel()):
    ax.imshow(digits.images[i], cmap='gray') # Mostrar la imagen en escala de grises
    ax.set_title(f'Label: {digits.target[i]}') # Título con la etiqueta del dígito
    ax.axis('off') # Quitar los ejes

plt.tight_layout()
plt.show()

```



Aplicar estadística descriptiva y extraer conclusiones

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits

# Cargar el dataset
digits = load_digits()

# Crear un DataFrame con los datos
df = pd.DataFrame(data=digits.data, columns=digits.feature_names)
df['target'] = digits.target

# Estadísticas descriptivas
print("Estadísticas descriptivas del dataset:")
print(df.describe())

# Visualización de la distribución de las etiquetas
plt.figure(figsize=(8, 5))
sns.countplot(x='target', data=df, palette='viridis')
plt.title('Distribución de las Etiquetas')
plt.xlabel('Etiqueta')
plt.ylabel('Número de Imágenes')
plt.show()

# Visualización de algunas imágenes de dígitos
def plot_digit_examples(images, labels, num_examples=5):
    plt.figure(figsize=(10, 5))
    for i in range(num_examples):
        plt.subplot(1, num_examples, i + 1)
        plt.imshow(images[i].reshape(8, 8), cmap='gray')
        plt.title(f'Label: {labels[i]}')
        plt.axis('off')
    plt.show()

# Mostrar ejemplos de imágenes
plot_digit_examples(digits.images, digits.target, num_examples=5)
```

Estadísticas descriptivas del dataset:

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	\
count	1797.0	1797.000000	1797.000000	1797.000000	1797.000000	
mean	0.0	0.303840	5.204786	11.835838	11.848080	
std	0.0	0.907192	4.754826	4.248842	4.287388	
min	0.0	0.000000	0.000000	0.000000	0.000000	
25%	0.0	0.000000	1.000000	10.000000	10.000000	
50%	0.0	0.000000	4.000000	13.000000	13.000000	
75%	0.0	0.000000	9.000000	15.000000	15.000000	
max	0.0	8.000000	16.000000	16.000000	16.000000	

	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	\
count	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	...	
mean	5.781859	1.362270	0.129661	0.005565	1.993879	...	
std	5.666418	3.325775	1.037383	0.094222	3.196160	...	
min	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	4.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	11.000000	0.000000	0.000000	0.000000	3.000000	...	
max	16.000000	16.000000	15.000000	2.000000	16.000000	...	

	pixel_6_7	pixel_7_0	pixel_7_1	pixel_7_2	pixel_7_3	\
count	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	
mean	0.206455	0.000556	0.279354	5.557596	12.089037	
std	0.984401	0.023590	0.934302	5.103019	4.374694	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	1.000000	11.000000	
50%	0.000000	0.000000	0.000000	4.000000	13.000000	
75%	0.000000	0.000000	0.000000	10.000000	16.000000	
max	13.000000	1.000000	9.000000	16.000000	16.000000	

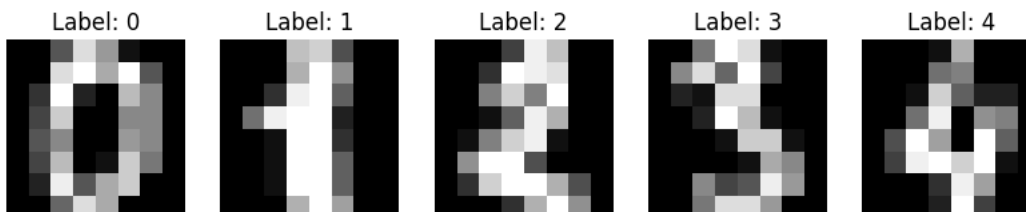
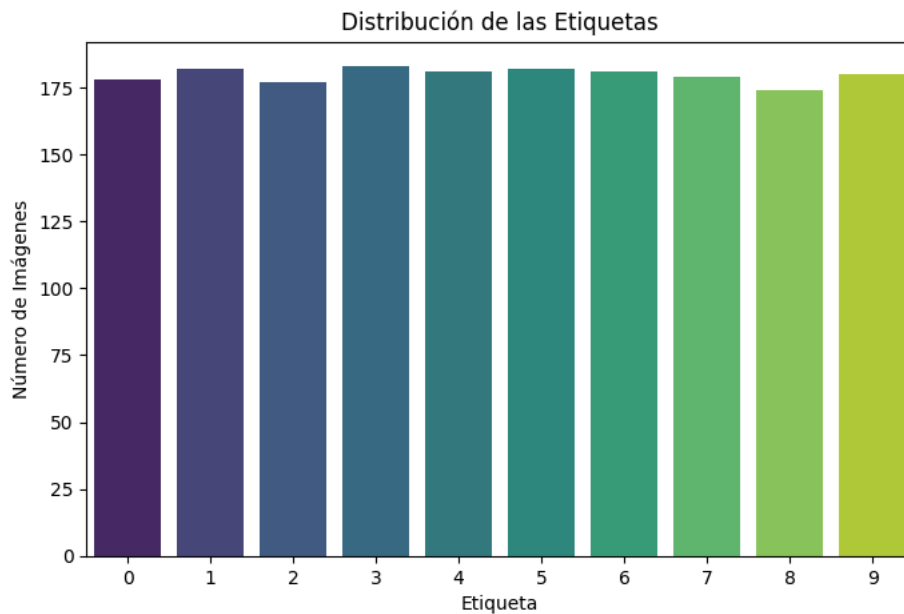
	pixel_7_4	pixel_7_5	pixel_7_6	pixel_7_7	target
count	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000
mean	11.809126	6.764051	2.067891	0.364496	4.490818
std	4.933947	5.900623	4.090548	1.860122	2.865304
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	10.000000	0.000000	0.000000	0.000000	2.000000
50%	14.000000	6.000000	0.000000	0.000000	4.000000
75%	16.000000	12.000000	2.000000	0.000000	7.000000
max	16.000000	16.000000	16.000000	16.000000	9.000000

[8 rows x 65 columns]

<ipython-input-11-dea38482c766>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

`sns.countplot(x='target', data=df, palette='viridis')`



Visualización

-Boxplot de algunas variables -Scatter plot entre algunas variables

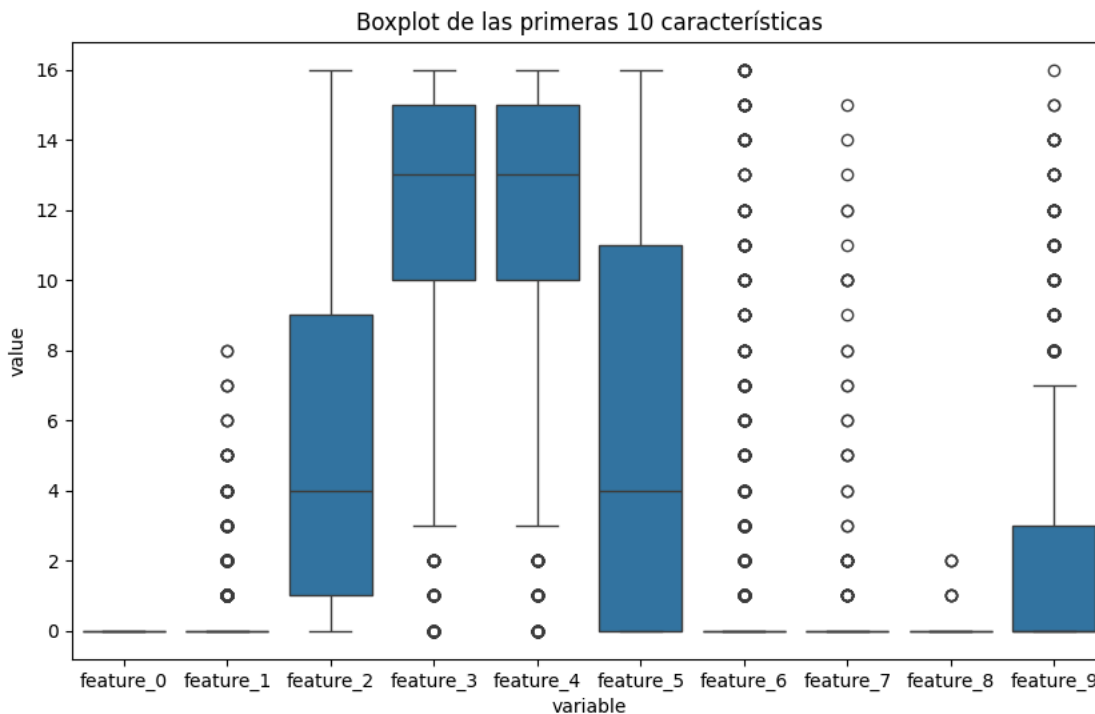
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits

# Cargar el dataset de dígitos
data = load_digits()

# Crear un DataFrame con los datos
df = pd.DataFrame(data.data, columns=[f'feature_{i}' for i in range(data.data.shape[1])])

# Seleccionar algunas columnas y reorganizar para seaborn
df_melted = df.iloc[:, :10].melt()

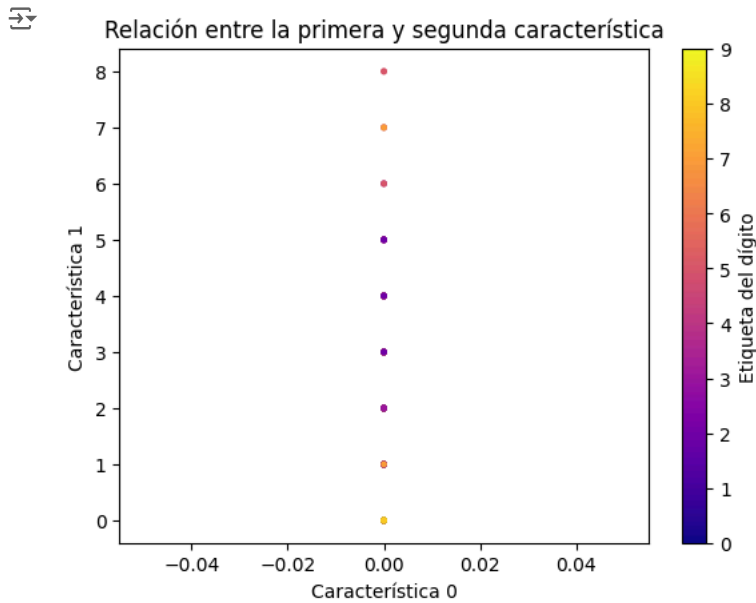
# Generar el boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(x='variable', y='value', data=df_melted)
plt.title("Boxplot de las primeras 10 características")
plt.show()
```



Scatter plot

```
import matplotlib.pyplot as plt

# Crear scatter plot entre las primeras dos características
plt.scatter(df.iloc[:, 0], df.iloc[:, 1], c=data.target, cmap='plasma', s=6)
plt.xlabel('Característica 0')
plt.ylabel('Característica 1')
plt.title("Relación entre la primera y segunda característica")
plt.colorbar(label='Etiqueta del dígito')
plt.show()
```



Do K-means con todas las variables -Revisar resultados

```
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, confusion_matrix
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import mode
from sklearn import datasets

# Cargar el dataset
digits_data = datasets.load_digits()

# Aplicar K-means usando todas las características disponibles (64)
kmeans_model = KMeans(n_clusters=10, random_state=42)
cluster_labels = kmeans_model.fit_predict(digits_data.data)

# Ajustar las etiquetas de los clusters basadas en la etiqueta predominante
final_labels = np.zeros(cluster_labels.shape)
for cluster_num in range(10):
    cluster_members = (cluster_labels == cluster_num)
    final_labels[cluster_members] = mode(digits_data.target[cluster_members])[0]

# Calcular la precisión
clustering_accuracy = accuracy_score(digits_data.target, final_labels)
print(f"Precisión del clustering usando todas las características: {clustering_accuracy:.2f}")

# Generar y mostrar la matriz de confusión
conf_matrix = confusion_matrix(digits_data.target, final_labels)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="YlGnBu", xticklabels=digits_data.target_names, yticklabels=digits_data.target_names)
plt.xlabel("Etiqueta predicha")
plt.ylabel("Etiqueta real")
plt.title("Matriz de Confusión: K-means con Características Completas")
plt.show()
```

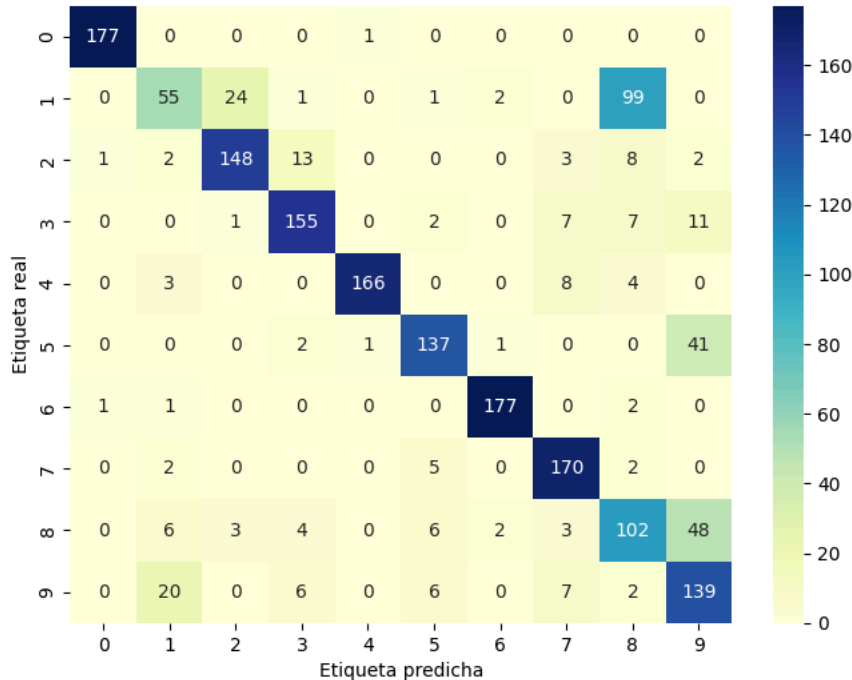
```

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will c
super()._check_params_vs_input(X, default_n_init=10)

```

Precisión del clustering usando todas las características: 0.79

Matriz de Confusión: K-means con Características Completas



Do K-means con solo variables de una de las filas de cada image -con que filas se hace bien o mal

```

# Función personalizada para seleccionar una fila específica de cada imagen
def extract_row(image_data, row_idx):
    return np.array([image[row_idx, :] for image in image_data])

# Fila a utilizar
row_num = 7
row_selected_data = extract_row(dig_data.images, row_num).reshape(len(dig_data.images), -1)


# Aplicar K-means sobre la fila seleccionada
kmeans_row_based = KMeans(n_clusters=10, random_state=42)
row_cluster_labels = kmeans_row_based.fit_predict(row_selected_data)

# Ajustar las etiquetas según la mayoría
majority_labels = np.zeros_like(row_cluster_labels)
for cluster in range(10):
    in_this_cluster = (row_cluster_labels == cluster)
    majority_labels[in_this_cluster] = mode(dig_data.target[in_this_cluster])[0]

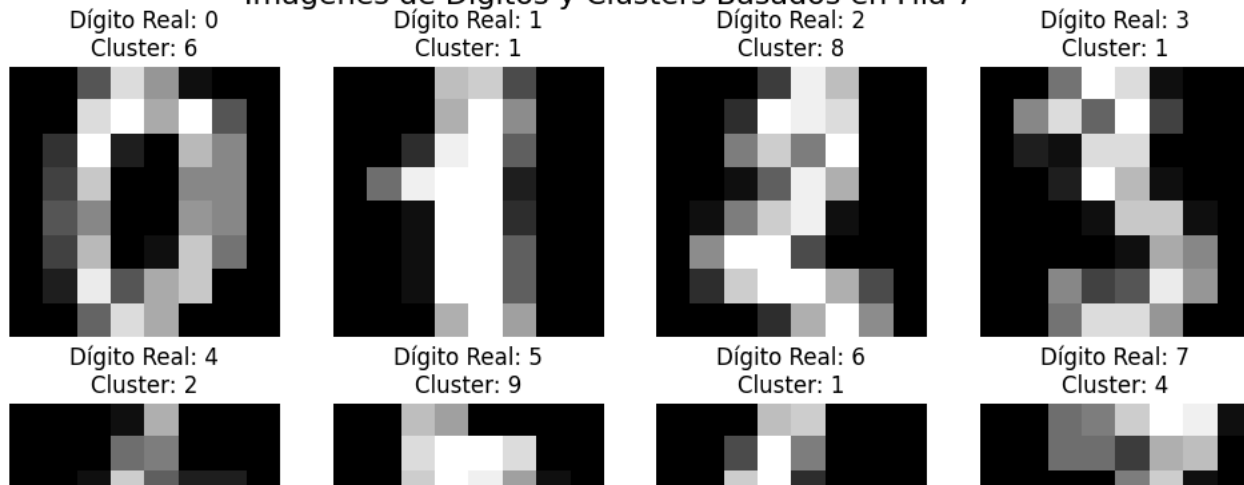
# Mostrar las primeras 8 imágenes con sus etiquetas de cluster asignadas
plt.figure(figsize=(12, 6))
for index in range(8):
    plt.subplot(2, 4, index + 1)
    plt.imshow(dig_data.images[index], cmap='gray')
    plt.title(f'Dígito Real: {dig_data.target[index]}\nCluster: {row_cluster_labels[index]}')
    plt.axis('off')

plt.suptitle(f"Imágenes de Dígitos y Clusters Basados en Fila {row_num}", fontsize=16)
plt.show()

```

 /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will c
super()._check_params_vs_input(X, default_n_init=10)

Imágenes de Dígitos y Clusters Basados en Fila 7



Do K-means con solo variables de una de las columnas de la imagen con que columnas se hace bien o mal



```
# Función para extraer una columna específica de cada imagen
def extract_column(image_data, col_idx):
    return np.array([image[:, col_idx] for image in image_data])


# Columna a utilizar
col_num = 5
col_selected_data = extract_column(dig_data.images, col_num).reshape(len(dig_data.images), -1)

# Aplicar K-means sobre la columna seleccionada
kmeans_col_based = KMeans(n_clusters=10, random_state=42)
col_cluster_labels = kmeans_col_based.fit_predict(col_selected_data)

# Ajustar las etiquetas de los clusters según la mayoría
majority_col_labels = np.zeros_like(col_cluster_labels)
for cluster in range(10):
    in_this_col_cluster = (col_cluster_labels == cluster)
    majority_col_labels[in_this_col_cluster] = mode(dig_data.target[in_this_col_cluster])[0]

# Visualizar las primeras 8 imágenes originales con los clusters asignados
plt.figure(figsize=(12, 6))
for idx in range(8):
    plt.subplot(2, 4, idx + 1)
    plt.imshow(dig_data.images[idx], cmap='gray')
    plt.title(f'Dígito Real: {dig_data.target[idx]}\nCluster: {col_cluster_labels[idx]}')
    plt.axis('off')

plt.suptitle(f"Imágenes de Dígitos y Clusters Basados en Columna {col_num}", fontsize=16)
plt.show()
```

 /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will c