

Importing Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Data

```
from sklearn.datasets import load_digits
digits = load_digits()
df = pd.DataFrame(digits.data)
```

Entender los dataset

df

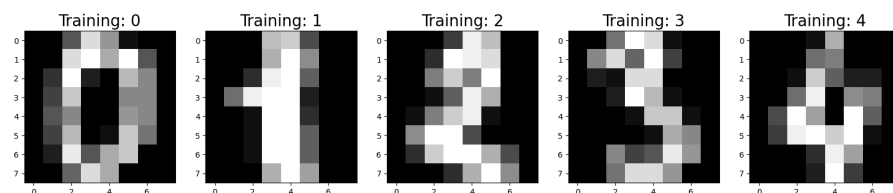


	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	7.0	13.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2.0
...
1792	0.0	0.0	4.0	10.0	13.0	6.0	0.0	0.0	0.0	1.0	...	4.0	0.0	0.0	0.0	2.0	14.0
1793	0.0	0.0	6.0	16.0	13.0	11.0	1.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	6.0	16.0
1794	0.0	0.0	1.0	11.0	15.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	9.0
1795	0.0	0.0	2.0	10.0	7.0	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0	5.0	12.0
1796	0.0	0.0	10.0	14.0	8.0	1.0	0.0	0.0	0.0	2.0	...	8.0	0.0	0.0	1.0	8.0	12.0

1797 rows × 64 columns

Plotear algunos ejemplos (digits)

```
plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.images[0:5], digits.target[0:5])):
    plt.subplot(1, 5, index + 1)
    plt.imshow(image, cmap=plt.cm.gray)
    plt.title('Training: %i' % label, fontsize = 20)
```



Aplicar estadística descriptiva y Extraer conclusiones

```
df.describe()
```



	0	1	2	3	4	5	6
count	1797.0	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000
mean	0.0	0.303840	5.204786	11.835838	11.848080	5.781859	1.362270
std	0.0	0.907192	4.754826	4.248842	4.287388	5.666418	3.325775
min	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.0	0.000000	1.000000	10.000000	10.000000	0.000000	0.000000
50%	0.0	0.000000	4.000000	13.000000	13.000000	4.000000	0.000000
75%	0.0	0.000000	9.000000	15.000000	15.000000	11.000000	0.000000
max	0.0	8.000000	16.000000	16.000000	16.000000	16.000000	16.000000

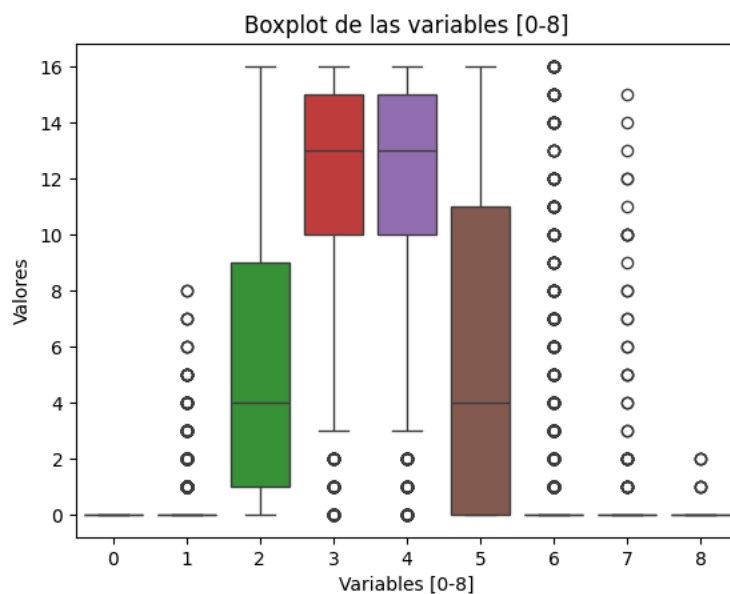
8 rows × 64 columns

Visualización:

- Boxplot de algunas variables
- Scatter Plot entre algunas variables

```
df_variables = df.iloc[:, [0,1,2,3,4,5,6,7,8]] # Select columns 4 and 5 using iloc
```

```
sns.boxplot(data=df_variables) # Use df_variables as the data source
plt.title('Boxplot de las variables [0-8]')
plt.xlabel('Variables [0-8]')
plt.ylabel('Valores')
plt.show()
```



```
# Select columns 0 to 8
selected_columns = df.iloc[:, :8]

# Plot the scatter plot
plt.figure(figsize=(10, 8))

# Create scatter plots for each pair of the first 8 columns
pd.plotting.scatter_matrix(selected_columns, alpha=0.8, figsize=(12, 12), diagonal='hist')

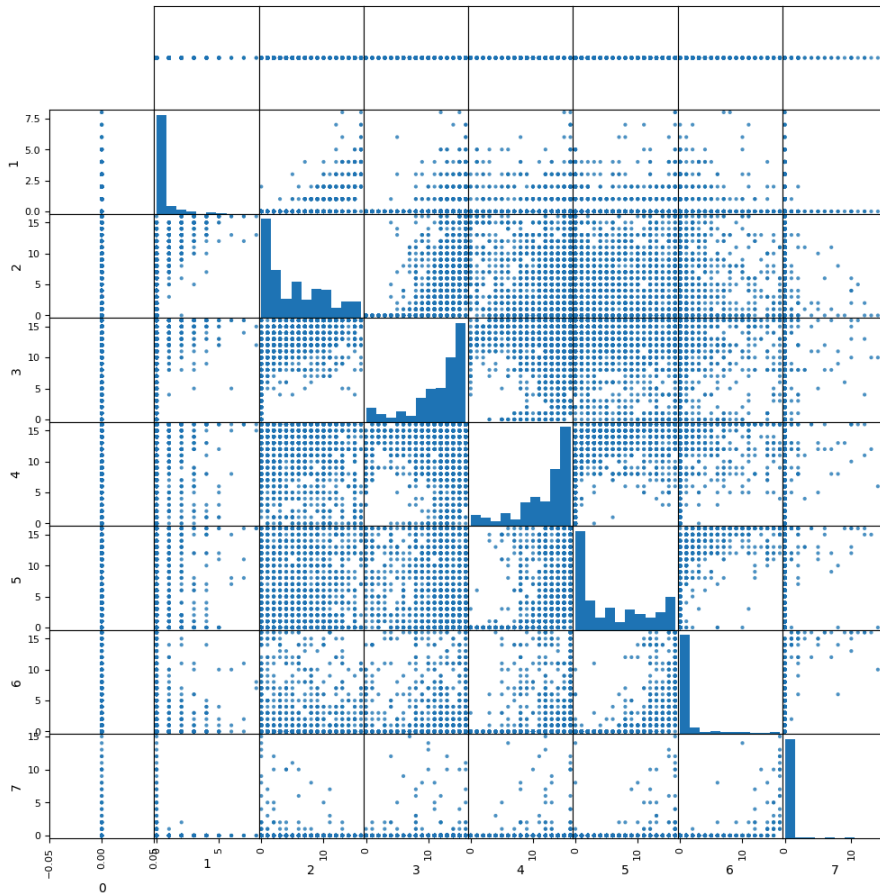
plt.suptitle('Scatter Plot for Digits Dataset (Columns 0 to 8)', fontsize=16)
plt.show()
```

```

/usr/local/lib/python3.10/dist-packages/pandas/plotting/_matplotlib/misc.py:91: UserWarning:
ax.set_xlim(boundaries_list[i])
/usr/local/lib/python3.10/dist-packages/pandas/plotting/_matplotlib/misc.py:101: UserWarning:
ax.set_ylim(boundaries_list[i])
/usr/local/lib/python3.10/dist-packages/pandas/plotting/_matplotlib/misc.py:100: UserWarning:
ax.set_xlim(boundaries_list[j])
/usr/local/lib/python3.10/dist-packages/pandas/plotting/_matplotlib/misc.py:115: RuntimeWarning:
adj = (locs - lim1[0]) / (lim1[1] - lim1[0])
<Figure size 1000x800 with 0 Axes>
WARNING:matplotlib.text:posx and posy should be finite values
WARNING:matplotlib.text:posx and posy should be finite values

```

Scatter Plot for Digits Dataset (Columns 0 to 8)



Do K-means con todas las variables:

- Revisar resultados

```


from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
# Extract features (64 columns) and target labels
X = digits.data

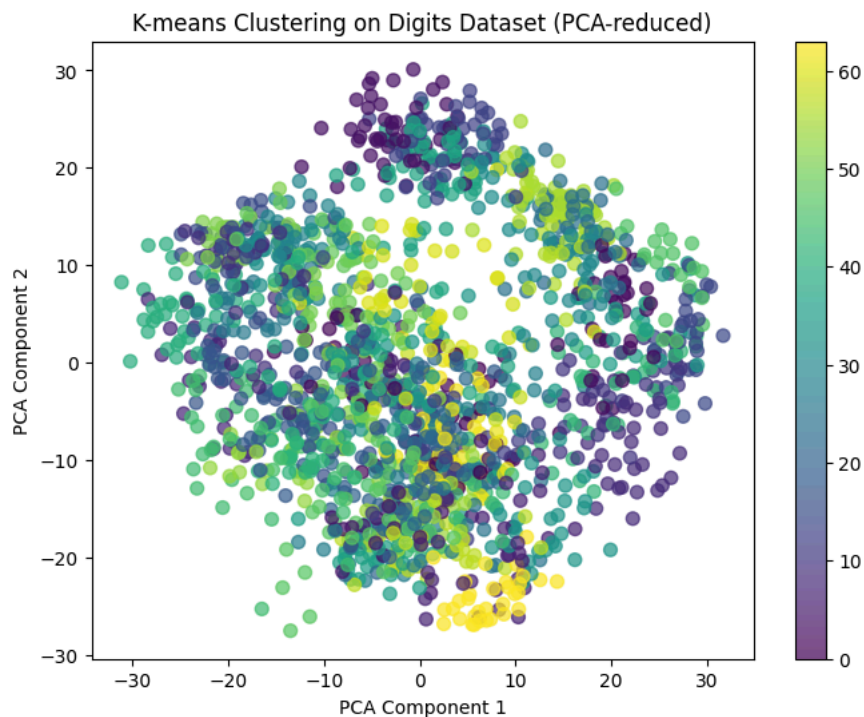
# Apply KMeans clustering
kmeans = KMeans(n_clusters=64, random_state=42) # 10 clusters for 10 digits (0-9)
clusters = kmeans.fit_predict(X)

# Reduce dimensionality for visualization using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Scatter plot of the clusters
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap='viridis', s=50, alpha=0.7)
plt.colorbar()
plt.title('K-means Clustering on Digits Dataset (PCA-reduced)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()

```

 /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: super()._check_params_vs_input(X, default_n_init=10)



Do k-means con solo variables de una de las filas de c/imagen:

- Con que filas se hace bien o mal

```

# Select one row from each image, e.g., the middle row (row index 4)
selected_row = 4
data = digits.images[:, selected_row, :]

# Reshape the data to be in the form (n_samples, n_features)
n_samples, n_features = data.shape

# Create and fit the k-means model
k = 10 # Number of clusters
kmeans = KMeans(n_clusters=k, random_state=0)
kmeans.fit(data)

# Predict the clusters
predicted_clusters = kmeans.predict(data)

# Plot the clusters
fig, ax = plt.subplots(2, 5, figsize=(10, 5))

```

```

for cluster in range(k):
    # Get the images belonging to the cluster
    cluster_images = digits.images[predicted_clusters == cluster]

    # Plot a few examples from each cluster
    for i in range(min(len(cluster_images), 5)):
        ax[cluster // 5, i].imshow(cluster_images[i], cmap='gray')
        ax[cluster // 5, i].axis('off')

plt.show()

```

 /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1 to 10. You should set `n_init` to the number of iterations desired.



Do k-means con solo variables de una de las columnas de la imagen:

- Con que columnas se hace bien o mal

```

# Select one column from each image, e.g., the middle column (column index 4)
selected_column = 4
data = digits.images[:, :, selected_column]

# Reshape the data to be in the form (n_samples, n_features)
n_samples, n_features = data.shape

# Create and fit the k-means model
k = 10 # Number of clusters
kmeans = KMeans(n_clusters=k, random_state=0)
kmeans.fit(data)

# Predict the clusters
predicted_clusters = kmeans.predict(data)

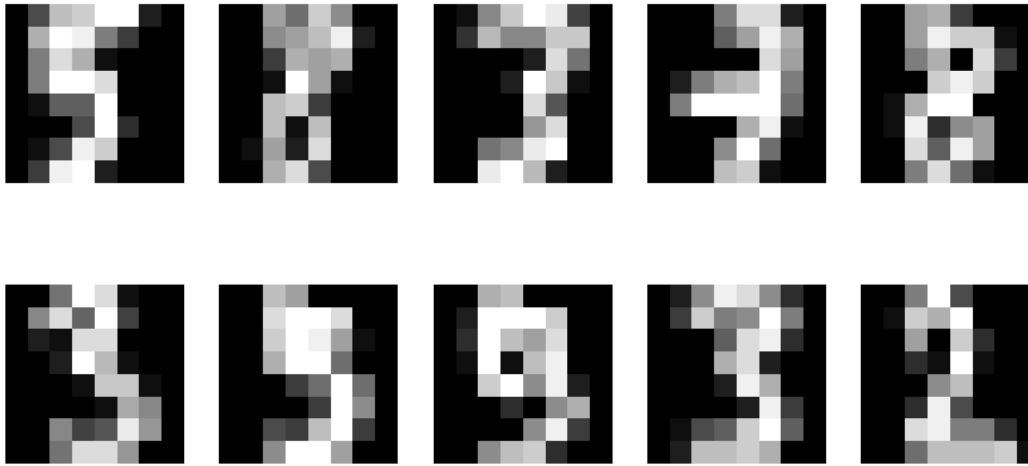
# Plot the clusters
fig, ax = plt.subplots(2, 5, figsize=(10, 5))
for cluster in range(k):
    # Get the images belonging to the cluster
    cluster_images = digits.images[predicted_clusters == cluster]

    # Plot a few examples from each cluster
    for i in range(min(len(cluster_images), 5)):
        ax[cluster // 5, i].imshow(cluster_images[i], cmap='gray')
        ax[cluster // 5, i].axis('off')

plt.show()

```

 /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 1 to 10 in version 1.3. To suppress this warning, you can explicitly pass `n_init=10` to the function.



Conclusión

Este análisis del conjunto de datos de "digits" examina imágenes de dígitos escritas a mano. Utilizando PCA, K-means y técnicas de reducción de dimensionalidad, se revelaron grupos de dígitos. Además, las pruebas que involucraron filas y columnas de imágenes particulares muestran que algunas características visuales son más beneficiosas para identificar números.

No se ha podido establecer conexión con el servicio reCAPTCHA. Comprueba tu conexión a Internet y vuelve a cargar la página para ver otro reCAPTCHA.