K-Vecinos más cercanos

```
Versión   : 1.2, Para uso educativo
Autor     : Luis Beltran Palma Ttito
Lugar     : Cusco, Perú, 2023.
Proposito : K-NN en la clasificación de digitos manuscritos
```

1. PASO PREVIOS

Importanción de librerías

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import random

# Librería: scikit learn, "sklearn"

# Módulo train_test_split: separación de datos para entrenamiento y test
from sklearn.model_selection import train_test_split
# Módulo KNeighborsClassifier: Implementa K-NN para clasificación
from sklearn.neighbors import KNeighborsClassifier
# Módulo classification_report: Métricas
from sklearn.metrics import classification_report
# Módulo accuracy_score: metrica de exactitud
from sklearn.metrics import accuracy_score
# Módulo confusion_matrix: matriz de confusión
from sklearn.metrics import confusion_matrix
```

Importación de datos

```
# leer el dataset
datos = pd.read_csv("digito16x16.csv", sep=',', header=0)
# muestras los primero 5 valores
datos.head()
```

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	...	p248	p249	p250	p251	p252	p253	p254	p255	p256	digito
0	0	0	0	0	0	0	1	1	1	1	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	1	1	1	1	...	1	1	1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	1	1	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0

5 rows × 257 columns

Distribución de clases

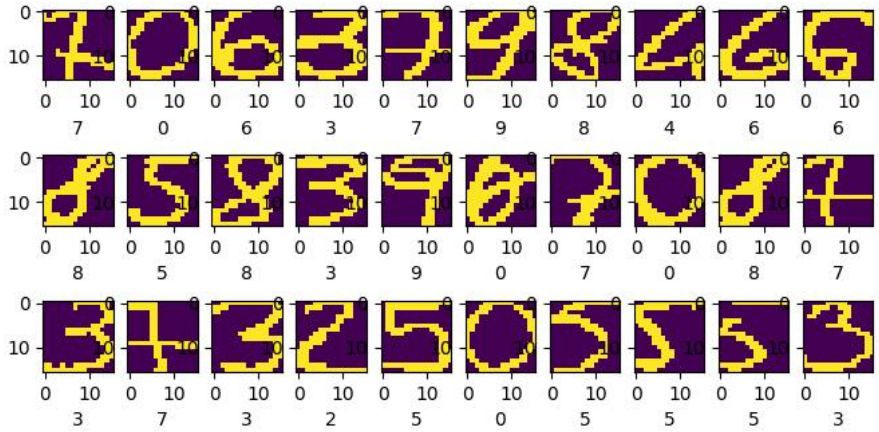
```
# mostrar distribución de clases mediante un gráfico de pye
distribucion = datos.groupby('digito').size()
plt.pie(distribucion, labels = ['cero','uno','dos', 'tres','cuatro','cinco','seis','siete','ocho','nueve'], autopct="%0.1f %%")
plt.title('Distribución de clases')
plt.show()

```

Visualización de datos

```
#visualizar algunos datos aleatoriamente
X = datos.drop(['digito'], axis=1)
y = datos['digito']

ran = datos.shape[0]-1
fig, ax = plt.subplots(3, 10, figsize=(8,4))
for k in range(3):
    for j in range(10):
        azar = random.sample(range(ran),1)
        img = np.array(X.iloc[azar].values)
        img = np.array(img).reshape(16,16)
        ax[k,j].imshow(img)
        ax[k,j].set_xlabel(y[azar[0]])
plt.show()
```



2. SEPARACIÓN DE DATOS PARA TRAINIG Y TESTING

```
# Separación de datos en entrada y salida
X = datos.drop(['digito'], axis=1)
y = datos['digito']

# Separación de datos para entrenamiento y test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state=5)
```

3. Entrenamiento y Test de K-NN

```
# Definición del modelo
Modelo = KNeighborsClassifier(n_neighbors = 5)
# Entrenamiento del modelo
Modelo.fit(X_train, y_train)
# Test del modelo
Exactitud = Modelo.score(X_test, y_test)
print('Exactitud de K-NN, k=3, :', Exactitud)
```

```
Exactitud de K-NN, k=3, : 0.891213389121339
```

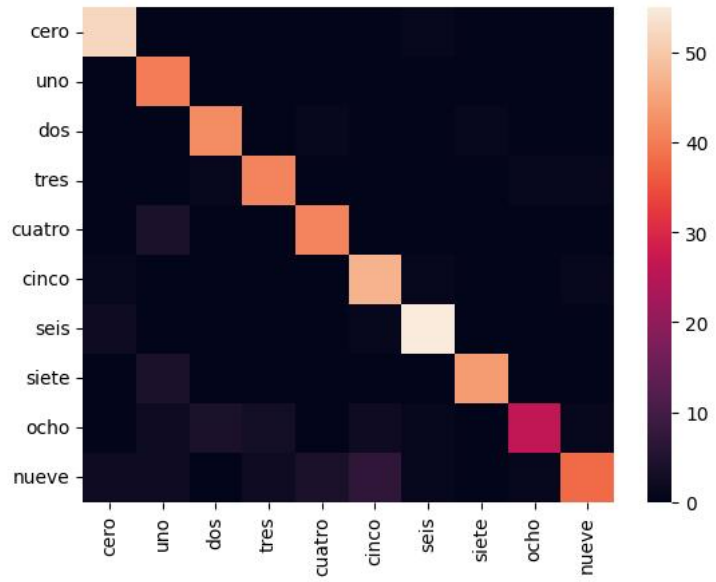
Matriz de confusión y exactitud

```
# Test del modelo
pred = Modelo.predict(X_test)
# Matriz de confusión
matriz = confusion_matrix(y_test, pred)
print(matriz)
```

```
[[52  0  0  0  0  0  1  0  0  0]
 [ 0 40  0  0  0  0  0  0  0  0]
 [ 0  0 42  0  1  0  0  1  0  0]
 [ 0  0  1 41  0  0  0  0  1  1]
 [ 0  4  0  0 41  0  0  0  0  0]
 [ 1  0  0  0  0 47  1  0  0  1]
 [ 2  0  0  0  0  1 55  0  0  0]
 [ 0  4  0  0  0  0  0 44  0  0]
 [ 0  2  4  3  0  2  1  0 26  1]
 [ 2  2  0  2  4  7  1  0  1 38]]
```

```
#visualización de la matriz de confusión como un mapa de calor
categorias = ['cero','uno','dos', 'tres','cuatro','cinco','seis','siete','ocho','nueve']
sns.heatmap(matriz, yticklabels=categorias, xticklabels=categorias)
```

```
<Axes: >
```




```
from sklearn.metrics import accuracy_score
# Cálculo de acurracy (exactitud)
accuracy = accuracy_score(y_test, pred)
print('Exactitud: ',accuracy)
```

```
Exactitud:  0.891213389121339
```

4. PREDICCIÓN DE NUEVOS CASOS

```
nuevo = np.array([[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,
                   0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,1,1,
                   0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,
                   0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,1,0,
                   0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,
                   0,0,0,1,1,1,0,1,0,1,0,1,0,0,0,0,0,
                   0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,
                   0,0,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,
                   0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,0,
                   1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,0,
                   1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,
                   1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,
                   1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,
                   1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,
                   1,1,1,0,1,1,1,0,1,1,1,1,0,0,0,0,
                   0,1,1,1,1,1,1,0,0,0,1,0,0,0,0,0]])
```

```
1 y_pred = Modelo.predict(nuevo)
2 # mostrar el resultado de la clasificación del dígito
3 print('El dígito es: ',y_pred[0])
```

```
El dígito es:  6
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassi
  warnings.warn(
```

4. PRÁCTICA

Aplicar el algoritmo k-NN para los siguientes dataset

> winequality

> dureza.csv

De igual manera experimente con los datos vectorizados de tubérculos y corte de papas nativas del proyecto.

```
```