

Machine Learning- K-nearest neighbors (KNN)

https://www.w3schools.com/python/python_ml_knn.asp



ChatGPT

Este código en Python utiliza la biblioteca `matplotlib` para crear un gráfico de dispersión y visualizar un conjunto de datos bidimensional. Posteriormente, se aplica el algoritmo de vecinos más cercanos (KNN) para clasificar puntos en dos clases diferentes.

Vamos a desglosar el código:

1. Importar Librerías:

```
python Copy code  
  
import sys  
import matplotlib  
matplotlib.use('Agg')
```

- Estas líneas importan las bibliotecas necesarias. `sys` se utiliza para la manipulación de flujo de salida, y `matplotlib` es una biblioteca de visualización en Python.

2. Definir Datos de Entrada:

```
python Copy code  
  
x = [4, 5, 10, 4, 3, 11, 14, 8, 10, 12]  
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]  
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
```

- Se definen dos listas (`x` e `y`) que representan las coordenadas de los puntos en el gráfico de dispersión. La lista `classes` indica a qué clase pertenece cada punto (0 o 1).

3. Crear Gráfico de Dispersión:

```
python Copy code  
  
plt.scatter(x, y, c=classes)  
plt.show()
```

- Utilizando `matplotlib`, se crea un gráfico de dispersión (`scatter plot`) donde los puntos se colocan en las coordenadas definidas por `x` e `y`, y se colorean según la clase especificada en `classes`. `plt.show()` muestra el gráfico.

4. Guardar la Figura y Limpiar el Buffer:

```
python Copy code  
  
plt.savefig(sys.stdout.buffer)  
sys.stdout.flush()
```

- Esta parte del código guarda la figura en el flujo de salida estándar (`sys.stdout.buffer`). Esto es necesario cuando se ejecuta el código en entornos que no admiten la visualización directa de gráficos, como algunos entornos en línea. `sys.stdout.flush()` asegura que se limpie el búfer de salida.

En resumen, este código crea un gráfico de dispersión de puntos bidimensionales y los colorea según su clase. Puedes imaginarte que estos puntos representan datos en un espacio bidimensional, y podríamos querer aplicar el algoritmo KNN para clasificar nuevos puntos basándonos en su proximidad a los existentes en términos de distancia euclidiana. Este es solo el paso inicial, y el código podría extenderse para aplicar el algoritmo KNN y visualizar cómo clasifica nuevos puntos en el conjunto de datos.



ChatGPT

¡Por supuesto! Podemos agregar una utilidad al ejemplo implementando el algoritmo KNN para clasificar un nuevo punto en función de su proximidad a los puntos existentes en el conjunto de datos. Aquí hay una extensión del código para mostrar cómo podría hacerse:

python

Copy code

```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np

# Definir el nuevo punto que queremos clasificar
new_point = np.array([[6, 20]])

# Crear un clasificador KNN con k=3 (3 vecinos más cercanos)
knn_classifier = KNeighborsClassifier(n_neighbors=3)
# Entrenar el clasificador con los datos existentes
knn_classifier.fit(np.column_stack((x, y)), classes)

# Clasificar el nuevo punto
predicted_class = knn_classifier.predict(new_point)
print(f"La clase predicha para el nuevo punto {new_point} es: {predicted_class}")

# Crear un gráfico de dispersión con el nuevo punto coloreado
plt.scatter(x, y, c=classes)
plt.scatter(new_point[0, 0], new_point[0, 1], marker='X', c='red', label='New Point')
plt.legend()
plt.show()
```

En este código, hemos introducido un nuevo punto `[6, 20]` y utilizamos el clasificador KNN para predecir a qué clase pertenece. Luego, visualizamos el conjunto de datos original junto con el nuevo punto y su clasificación predicha. En este caso, el nuevo punto se marca con una "X" y se colorea de rojo.

Esta extensión muestra cómo el algoritmo KNN puede ser utilizado para realizar predicciones sobre nuevos datos en función de su proximidad a los datos existentes en el conjunto de entrenamiento. Este es un ejemplo básico, pero ilustra la idea general de clasificación con KNN.



ChatGPT

Claro, puedes modificar el código para guardar el gráfico como una imagen en formato PNG en lugar de mostrarlo directamente. Aquí está el código modificado:

python

Copy code

```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np

# Definir los datos
x = [4, 5, 10, 4, 3, 11, 14, 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

# Crear el clasificador KNN
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(np.column_stack((x, y)), classes)

# Definir el nuevo punto que queremos clasificar
new_point = np.array([[6, 20]])

# Clasificar el nuevo punto
predicted_class = knn_classifier.predict(new_point)
print(f"La clase predicha para el nuevo punto {new_point} es: {predicted_class}")

# Crear un gráfico de dispersión con el nuevo punto coloreado
plt.scatter(x, y, c=classes)
plt.scatter(new_point[0, 0], new_point[0, 1], marker='X', c='red', label='New Point')
plt.legend()

# Guardar el gráfico como una imagen en formato PNG
plt.savefig('scatter_plot.png')
```

En este código, se ha eliminado la línea `plt.show()` que mostraba el gráfico directamente. En su lugar, se ha añadido la línea `plt.savefig('scatter_plot.png')`, que guarda el gráfico como una imagen en formato PNG con el nombre de archivo "scatter_plot.png". Puedes ajustar el nombre del archivo según tus preferencias. Después de ejecutar este código, encontrarás el archivo PNG en el directorio donde se ejecutó el script.