KNN:

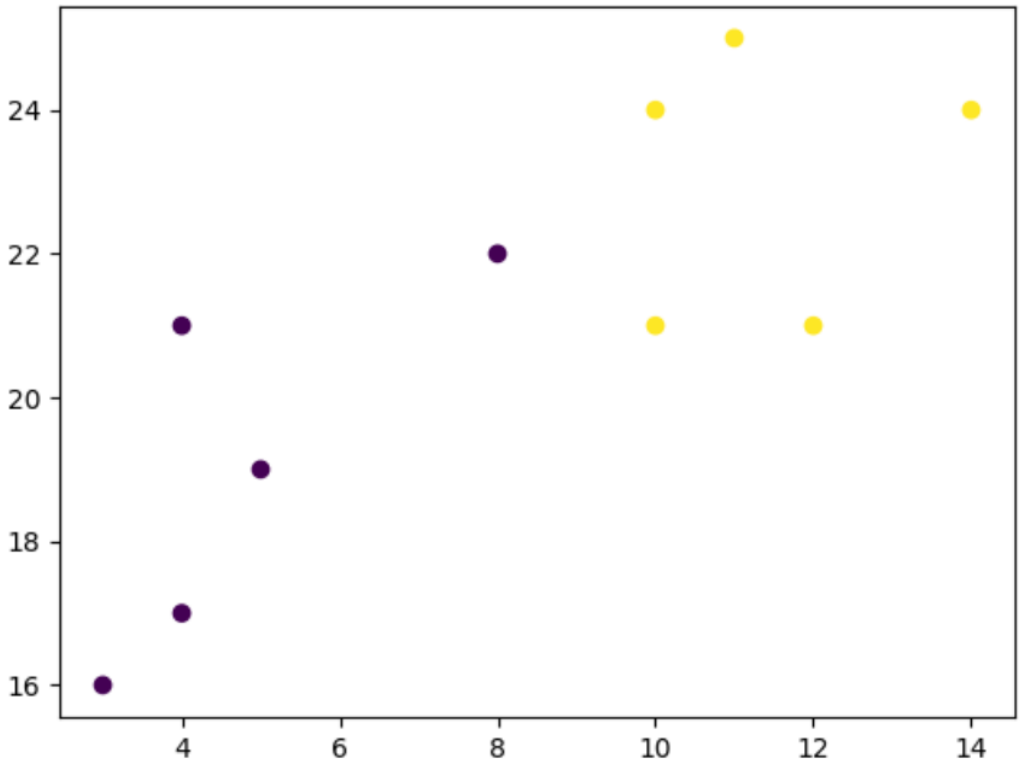
KNN, sınıflandırma veya regresyon görevleri için kullanılabilen basit, denetimli bir makine öğrenimi (ML) algoritmasıdır ve eksik değer atamasında da sıklıkla kullanılır. Belirli bir veri noktasına en yakın gözlemlerin bir veri kümesindeki en "benzer" gözlemler olduğu fikrine dayanır ve bu nedenle öngörülemeyen noktaları mevcut en yakın noktaların değerlerine göre sınıflandırabiliriz. K'yi seçerek, kullanıcı algoritmada kullanılacak yakındaki gözlemlerin sayısını seçebilir.

Bazı veri noktalarını görselleştirerek başlayın:

CODE:

```
import sys

import matplotlib matplotlib.use('Agg')

import matplotlib.pyplot as plt

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]

plt.scatter(x, y, c=classes)

plt.show()

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()
```

OUTPUT:



Şimdi KNN algoritmasını K=1 ile eşleştiriyoruz:

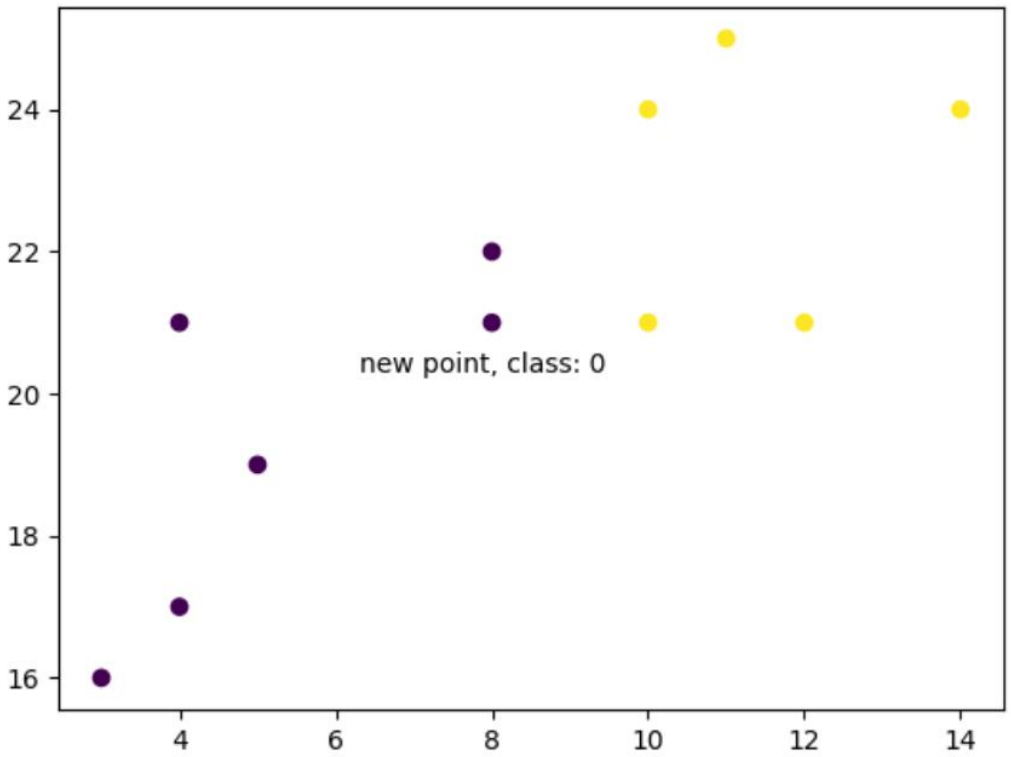
```
from sklearn.neighbors

import KNeighborsClassifier

data = list(zip(x, y))

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(data, classes)
```

Ve yeni bir veri noktasını sınıflandırmak için kullanın:

CODE:

```
import sys

import matplotlib matplotlib.use('Agg')

import matplotlib.pyplot as plt from sklearn.neighbors

import KNeighborsClassifier

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]

data = list(zip(x, y))

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(data, classes)

new_x = 8

new_y = 21

new_point = [(new_x, new_y)]

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])

plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")

plt.show()

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()
```

OUTPUT:



Şimdi aynı şeyi yapıyoruz, ancak tahmini değiştiren daha yüksek bir K değeriyle:

CODE:

```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(data, classes)

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])

plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")

plt.show()
```

OUTPUT: