marker\_shapes = ‘v^os’

mapper = [marker\_shapes[i] for i in y]

for i in range (X.shape[0]) :

plt.scatter(X[i, 0], X[i, 1], marker=mapper[i]),

s=75, edgecoor=’black’, facecolor=’none’)

Tetapkan jumlah tetangga terdekat yang ingin kita gunakan:

# Jumlah tetangga terdekat

num\_neighbors = 12

Buat model pengklasifikasi K Nearest Neighbors:

# Buat model pengklasifikasi K Nearest Neighbors

classifier = neighbors.KNeighborsClassifier(num\_neighbors,

weights='distance')1

Coba model menggunakan data percobaan:

# Coba model menggunakan data percobaan:

classifier.fit(X, y)

Buat grid mesh dari nilai yang akan digunakan untuk memvisualisasikan grid:

# buat mesh untuk merencanakan batas

x-min, x\_max = X[:, 0].min() – 1, X[:, 0].max() + 1

y-min, y\_max = X[:, 0].min() – 1, X[:, 0].max() + 1

x\_values, y\_values = np.meshgrid(np.arange(x\_min, x\_max, step\_size),np.arange (y\_min, y\_max, step\_size))

Evaluasilah classifier pada semua titik di grid untuk membuat sebuah visualisasi dari batas:

# Mengevaluasi classifier pada semua poin di grid

output = classifier.predict(np.c\_[x\_values.ravel(), y\_values.ravel()])

Buat mesh warna untuk memvisualisasikan output:

#Visualisasikan output yang diprediksi

output = output.reshape(x\_values.shape)

plt.figure()

plt.pcolormesh(x\_values, y\_values, output, cmap=cm.Paired)

Paparkan data pelatihan di atas mesh warna ini untuk memvisualisasikan data relatif terhadap batas:

# poin pelatihan di peta

for i in range(X.shape[0]):

plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],

s=50, edgecolors='black', facecolors='none')

Tetapkan batas X dan Y beserta judul:

plt.xlim(x\_values.min(), x\_values.max())

plt.ylim(y\_values.min(), y\_values.max())

plt.title('K Nearest Neighbors classifier model boundaries')

Tentukan titik data pengujian untuk melihat bagaimana kinerja pengklasifikasian. Buat gambar dengan titik data pelatihan dan titik data uji untuk melihat di mana letaknya:

# Test input data point

test\_datapoint = [5.1, 3.6]

plt.figure()

plt.title('Test datapoint')

for i in range(X.shape[0]):

plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],

s=75, edgecolors='black', facecolors='none')

plt.scatter(test\_datapoint[0], test\_datapoint[1], marker='x',

linewidth=6, s=200, facecolors='black')

Ekstrak Tetangga K Terdekat ke titik data uji, berdasarkan model classifier:

# Ekstrak K tetangga terdekat

\_, indices = classifier.kneighbors([test\_datapoint])

indices = indices.astype(np.int)[0]

Plot K tetangga terdekat yang diperoleh pada langkah sebelumnya:

# Plot k nearest neighbors

plt.figure()

plt.title('K Nearest Neighbors')

for i in indices:

plt.scatter(X[i, 0], X[i, 1], marker=mapper[y[i]],

linewidth=3, s=100, facecolors='black')

Overlay titik data uji:

plt.scatter(test\_datapoint[0], test\_datapoint[1], marker='x',

linewidth=6, s=200, facecolors='black')

Overlay data input:

for i in range(X.shape[0]):

plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],

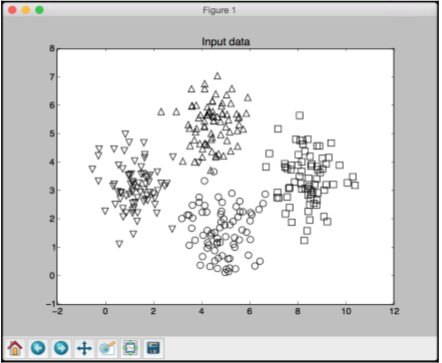
s=75, edgecolors='black', facecolors='none')

Cetak hasil prediksi:

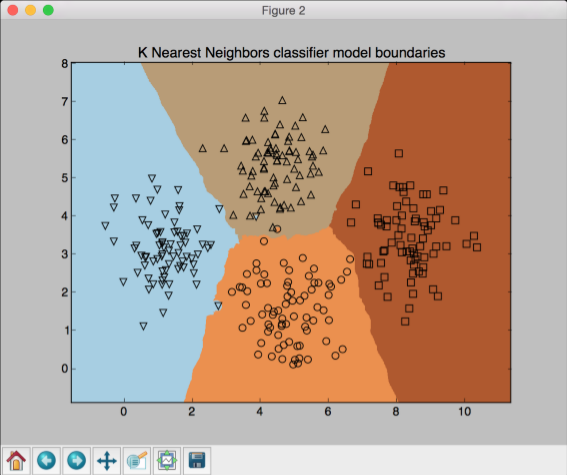
print("Predicted output:", classifier.predict([test\_datapoint])[0])

plt.show()

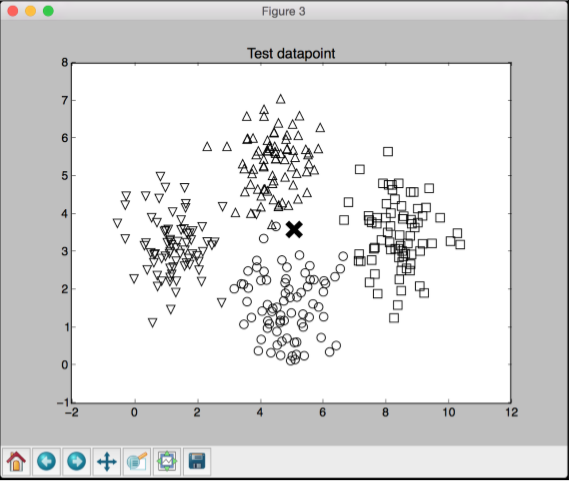
Kode lengkap diberikan dalam file nearest\_neighbors\_classifier.py. Jika Anda menjalankan kode, Anda akan melihat empat tangkapan layar. Tangkapan layar pertama menunjukkan data input:



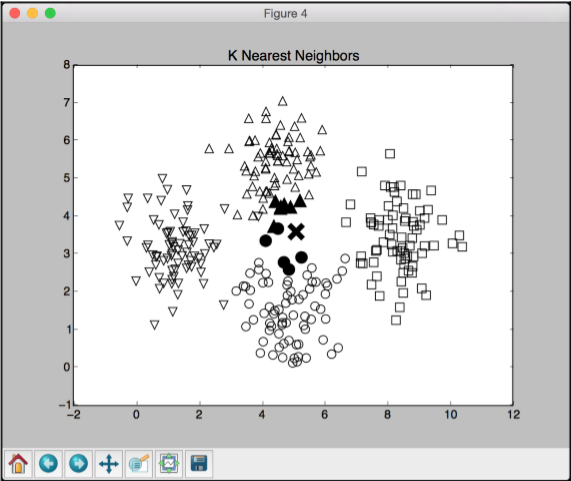
Tangkapan layar kedua mewakili batasan classifier:



Tangkapan layar ketiga menunjukkan titik data uji relatif terhadap dataset input. Titik data uji ditampilkan menggunakan tanda silang:



Tangkapan layar keempat menunjukkan 12 tetangga terdekat ke titik data uji:



Anda akan melihat output berikut di Terminal, menunjukkan titik data uji milik class 1:

Predicted output: 1

Computing similarity scores (Menghitung skor kesamaan)

Untuk membangun sistem rekomendasi, penting untuk memahami bagaimana membandingkan berbagai objek dalam dataset kita. Katakanlah dataset kita terdiri dari orang-orang dan berbagai preferensi film mereka. Untuk merekomendasikan sesuatu, kita perlu memahami bagaimana membandingkan dua orang dengan satu sama lain. Di sinilah similarity score menjadi sangat penting. similarity score memberi kita gambaran tentang betapa miripnya dua objek.

Ada dua skor yang sering digunakan dalam domain ini - skor Euclidean dan skor Pearson. **Skor Euclidean** menggunakan jarak Euclidean antara dua titik data untuk menghitung skor. Jika Anda memerlukan penyegaran cepat tentang bagaimana jarak Euclidean dihitung, Anda dapat mengunjungi https://en.wikipedia.org/wiki/Euclidean\_distance. Nilai jarak Euclidean bisa tidak terbatas. Oleh karena itu kami mengambil nilai ini dan mengubahnya dengan cara sehingga skor Euclidean berkisar dari 0 hingga 1. Jika jarak Euclidean antara dua objek besar, maka skor Euclidean harus rendah karena skor rendah menunjukkan bahwa objek tidak sama. **Karenanya jarak Euclidean berbanding terbalik dengan skor Euclidean.**

**Skor Pearson** adalah ukuran korelasi antara dua objek. Ia menggunakan kovarians antara dua objek bersama dengan penyimpangan standar masing-masing untuk menghitung skor. Skor dapat berkisar dari -1 hingga +1. Skor +1 menunjukkan bahwa objek sangat mirip sementara skor -1 akan menunjukkan bahwa objek sangat berbeda. Skor 0 akan menunjukkan bahwa tidak ada korelasi antara kedua objek. Mari kita lihat bagaimana cara menghitung skor ini.

Buat file Python baru dan impor paket-paket berikut:

import argparse

import json

import numpy as np

Buat parser argumen untuk memproses argumen input. Ini akan menerima dua pengguna dan jenis skor yang perlu digunakan untuk menghitung skor kesamaan:

def build\_arg\_parser():

parser = argparse.ArgumentParser(description='Compute similarity score')

parser.add\_argument('--user1', dest='user1', required=True, help='First user')

parser.add\_argument('--user2', dest='user2', required=True, help='Second

user')

parser.add\_argument("--score-type", dest="score\_type", required=True,

choices=['Euclidean', 'Pearson'], help='Similarity metric to be used')

return parser