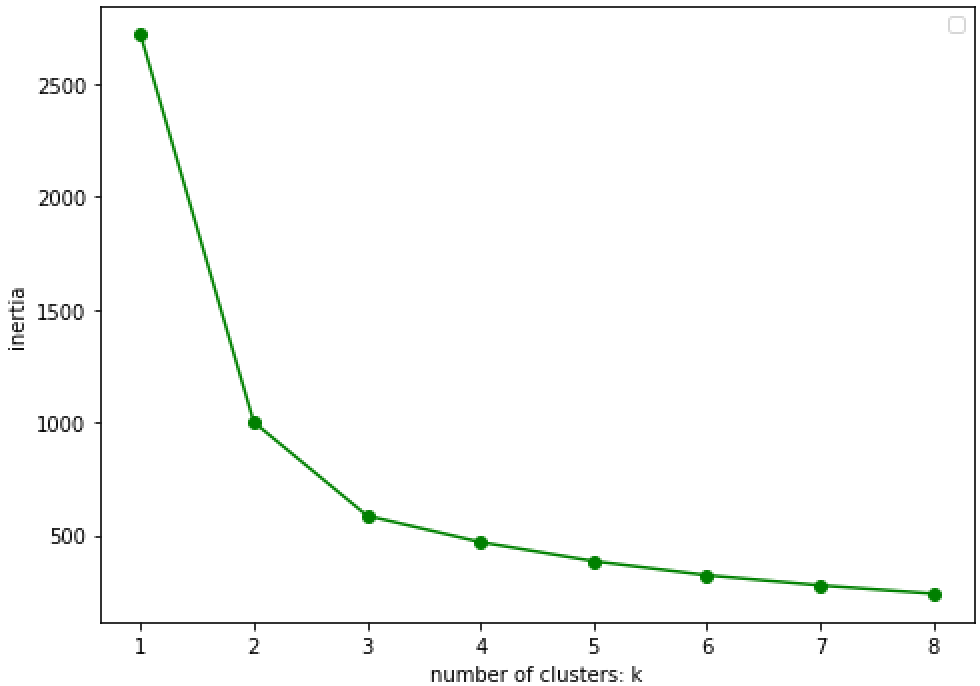
In [ ]:
```
"""
Jiankun Dong
Class: CS 677
Date: 11/20/2023
Q3 Origianl Dataset with all three classes cluster
"""
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from collections import Counter
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import random
import warnings
warnings.filterwarnings("ignore")

print("Part 1")
# R = 0: class L = 1 neg class L = 2 pos
data = pd.read_csv("seeds_dataset.csv",sep = '\t')
data.columns =  ["f1","f2","f3","f4","f5","f6","f7","L"]
featurelist = ["f1","f2","f3","f4","f5","f6","f7"]
X = data[featurelist].values
Y = data["L"].values
# not splitting data with this one
#X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=.5,random_
state=0)
colmap = {1: 'red', 2: 'green', 3:'blue',4:'grey',5:'purple',6:'yellow'}
kCount = [1,2,3,4,5,6,7,8]
inertia_list = []
for k in kCount:
    kmeans_classifier = KMeans(n_clusters=k)
    y_means = kmeans_classifier.fit_predict(X)
    centroids = kmeans_classifier.cluster_centers_
    inertia = kmeans_classifier.inertia_
    inertia_list.append(inertia)

fig,ax = plt.subplots(1,figsize =(7,5))
plt.plot(range(1, 9), inertia_list, marker='o',
        color='green')

plt.legend()
plt.xlabel('number of clusters: k')
plt.ylabel('inertia')
plt.tight_layout()

plt.show()
print("Part 2")
## the best k is 3
# rerun with k=3
kmeans_classifier = KMeans(n_clusters=3,random_state=0)
y_kmeans = kmeans_classifier.fit_predict(X)
data['Predict'] = y_kmeans
centroids = kmeans_classifier.cluster_centers_
# pick i and k at random
# The randomly picked fi and fk using the line below was
```

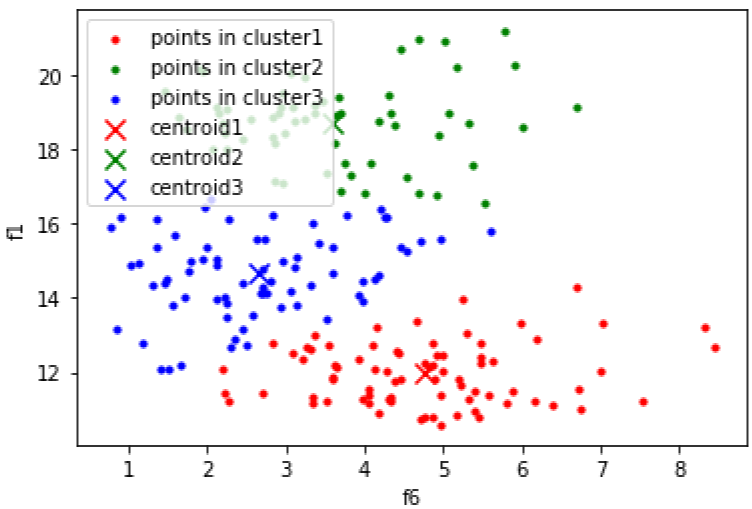
```
fi_index,fk_index = random.sample(range(0,7),2)
fi = featurelist[fi_index]
fk = featurelist[fk_index]
#plt.close()
fig = plt.figure()

for i in range (1,4):
    new_df = data[data['Predict']==i-1]
    plt.scatter(new_df[fi], new_df[fk], color=colmap[i],
                s=10 , label='points in cluster' + str(i))

for i in range (1,4):
    plt.scatter(centroids[i-1][fi_index], centroids[i-1][fk_index], color=colm
ap[i],
                marker='x', s=100, label='centroid' + str(i))


plt.legend(loc='upper left')

plt.xlabel(fi)
plt.ylabel(fk)
plt.show()

print("Part 3")
##assign each cluster with centeroid and label
# cluster 1
cluster_label = []
cluster= data[data['Predict']==0]['L']
value, count = Counter(cluster).most_common()[0]
cluster_label.append(value)
print("Assign cluster 0 with class label",value)
print("Centroid for cluster 0:",centroids[0])
cluster= data[data['Predict']==1]['L']
value, count = Counter(cluster).most_common()[0]
cluster_label.append(value)
print("Assign cluster 1 with class label",value)
print("Centroid for cluster 1:",centroids[1])
cluster= data[data['Predict']==2]['L']
value, count = Counter(cluster).most_common()[0]
cluster_label.append(value)
print("Assign cluster 2 with class label",value)
print("Centroid for cluster 2:",centroids[2])

fig = plt.figure()
for i in range (1,4):
    labelText = 'centroid' + str(i) + '(class'+str(cluster_label[i-1])+')'
    plt.scatter(centroids[i-1][fi_index], centroids[i-1][fk_index], color=colm
ap[i],
                marker='x', s=100, label=labelText)
plt.legend(fancybox=True,framealpha=0.5)
plt.xlabel(fi)
plt.ylabel(fk)
plt.show()

print("Part 4")
## np.linalg.norm() caculate the Euclidean distance between 2 points
distance_Class = []
```

```
for i in range(0,len(X)):
    dataPoint = X[i]
    disA = np.linalg.norm(dataPoint - centroids[0])
    disB = np.linalg.norm(dataPoint - centroids[1])
    disC = np.linalg.norm(dataPoint - centroids[2])
    distanceLS = [disA,disB,disC]
    minDis = min(distanceLS)
    if disA == minDis:
        #point closest to centroid 0
        distance_Class.append(cluster_label[0])
    elif disB == minDis:
        #point closest to centroid 1
        distance_Class.append(cluster_label[1])
    else:
        distance_Class.append(cluster_label[2])
data['DistanceClass'] = distance_Class
## Overall accuracy
accuracy = sum(data['L'] == data['DistanceClass'])/len(Y)
print("Overall accuracy is",accuracy)

print("Part 5")
##picking out result for label 1 and 2
new_df = data[data.L != 3]
accuracy = sum(new_df['L'] == new_df['DistanceClass'])/len(new_df.L)
print("Accuracy is:",accuracy)
## overvall confusion matrix
threeBythree = confusion_matrix(data.L.values,data.DistanceClass.values)
## getting rid of label 3
confusionMaxtrix = threeBythree[0:2,0:2]
tp = confusionMaxtrix[1,1]
fp = confusionMaxtrix[0,1]
tn = confusionMaxtrix[0,0]
fn = confusionMaxtrix[1,0]
print("tp: {0}, fp: {1}, tn: {2}, fn: {3}".format(tp, fp, tn, fn))
print("tpr: {0}, tnr: {1}".format(tp/(tp+fn),tn/(tn+fp)))
```

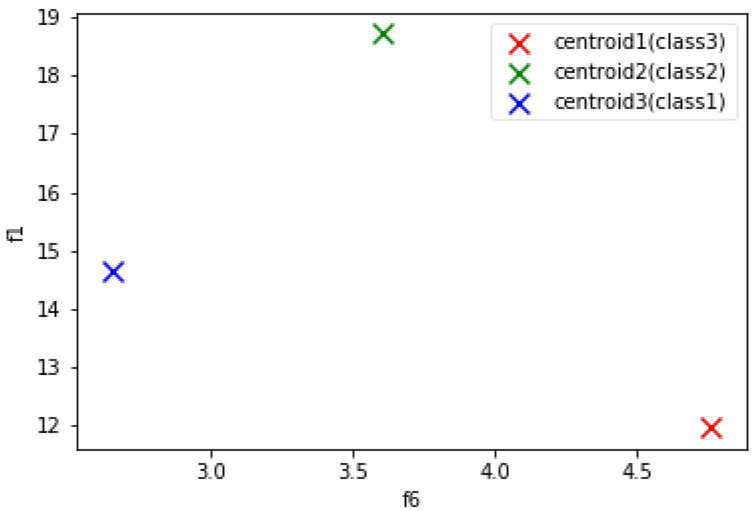
```
Part 1

No handles with labels found to put in legend.
```




```
Part 2
```




```
Part 3
Assign cluster 0 with class label 3
Centroid for cluster 0: [11.96441558 13.27480519  0.8522      5.22928571  2.8
7292208  4.75974026
  5.08851948]
Assign cluster 1 with class label 2
Centroid for cluster 1: [18.72180328 16.29737705  0.88508689  6.20893443  3.7
2267213  3.60359016
  6.06609836]
Assign cluster 2 with class label 1
Centroid for cluster 2: [14.63985915 14.45507042  0.87928169  5.56097183  3.2
7742254  2.65496056
  5.19192958]
```



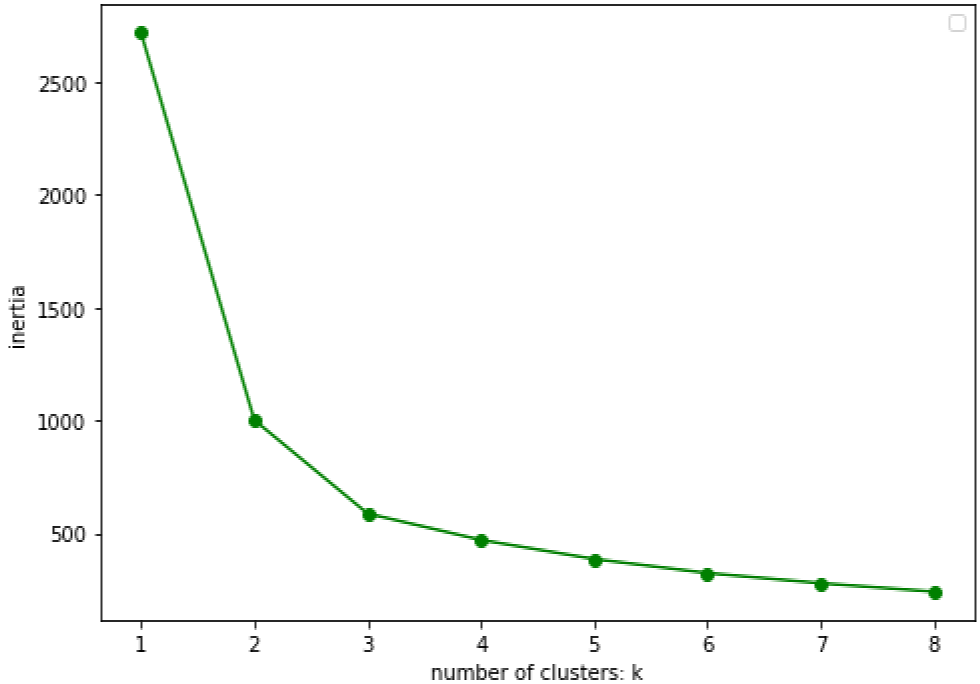

```
Part 4
Overall accuracy is 0.8947368421052632
Part 5
Accuracy is: 0.8561151079136691
tp: 60, fp: 1, tn: 59, fn: 10
tpr: 0.8571428571428571, tnr: 0.9833333333333333
```

In [ ]:
```
"""
Jiankun Dong
Class: CS 677
Date: 11/20/2023
Q3 Origianl Dataset with all three classes cluster
"""
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from collections import Counter
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import random
import warnings
warnings.filterwarnings("ignore")

print("Part 1")
# R = 0: class L = 1 neg class L = 2 pos
data = pd.read_csv("seeds_dataset.csv",sep = '\t')
data.columns =  ["f1","f2","f3","f4","f5","f6","f7","L"]
featurelist = ["f1","f2","f3","f4","f5","f6","f7"]
X = data[featurelist].values
Y = data["L"].values
# not splitting data with this one
#X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=.5,random_
state=0)
colmap = {1: 'red', 2: 'green', 3:'blue',4:'grey',5:'purple',6:'yellow'}
kCount = [1,2,3,4,5,6,7,8]
inertia_list = []
for k in kCount:
    kmeans_classifier = KMeans(n_clusters=k)
    y_means = kmeans_classifier.fit_predict(X)
    centroids = kmeans_classifier.cluster_centers_
    inertia = kmeans_classifier.inertia_
    inertia_list.append(inertia)

fig,ax = plt.subplots(1,figsize =(7,5))
plt.plot(range(1, 9), inertia_list, marker='o',
        color='green')

plt.legend()
plt.xlabel('number of clusters: k')
plt.ylabel('inertia')
plt.tight_layout()

plt.show()
print("Part 2")
## the best k is 3
# rerun with k=3
kmeans_classifier = KMeans(n_clusters=3,random_state=0)
y_kmeans = kmeans_classifier.fit_predict(X)
data['Predict'] = y_kmeans
centroids = kmeans_classifier.cluster_centers_
# pick i and k at random
# The randomly picked fi and fk using the line below was
```

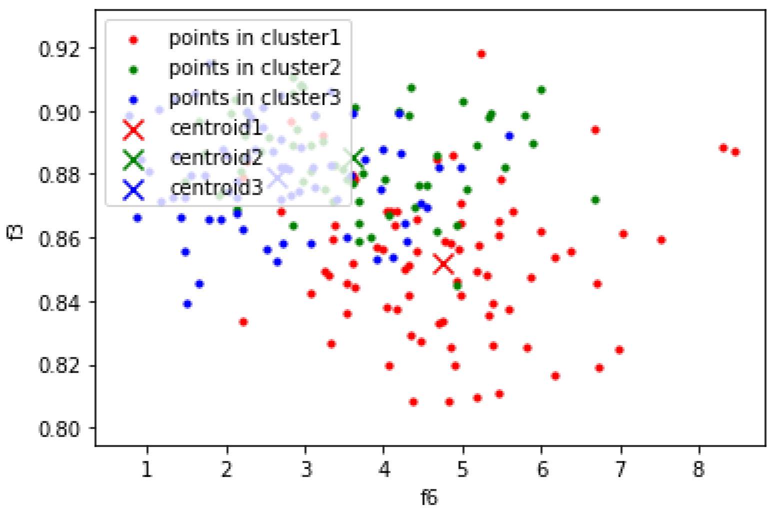
```
fi_index,fk_index = random.sample(range(0,7),2)
fi = featurelist[fi_index]
fk = featurelist[fk_index]
#plt.close()
fig = plt.figure()

for i in range (1,4):
    new_df = data[data['Predict']==i-1]
    plt.scatter(new_df[fi], new_df[fk], color=colmap[i],
                s=10 , label='points in cluster' + str(i))

for i in range (1,4):
    plt.scatter(centroids[i-1][fi_index], centroids[i-1][fk_index], color=colm
ap[i],
                marker='x', s=100, label='centroid' + str(i))


plt.legend(loc='upper left')

plt.xlabel(fi)
plt.ylabel(fk)
plt.show()

print("Part 3")
##assign each cluster with centeroid and label
# cluster 1
cluster_label = []
cluster= data[data['Predict']==0]['L']
value, count = Counter(cluster).most_common()[0]
cluster_label.append(value)
print("Assign cluster 0 with class label",value)
print("Centroid for cluster 0:",centroids[0])
cluster= data[data['Predict']==1]['L']
value, count = Counter(cluster).most_common()[0]
cluster_label.append(value)
print("Assign cluster 1 with class label",value)
print("Centroid for cluster 1:",centroids[1])
cluster= data[data['Predict']==2]['L']
value, count = Counter(cluster).most_common()[0]
cluster_label.append(value)
print("Assign cluster 2 with class label",value)
print("Centroid for cluster 2:",centroids[2])

fig = plt.figure()
for i in range (1,4):
    labelText = 'centroid' + str(i) + '(class'+str(cluster_label[i-1])+')'
    plt.scatter(centroids[i-1][fi_index], centroids[i-1][fk_index], color=colm
ap[i],
                marker='x', s=100, label=labelText)
plt.legend(fancybox=True,framealpha=0.5)
plt.xlabel(fi)
plt.ylabel(fk)
plt.show()

print("Part 4")
## np.linalg.norm() caculate the Euclidean distance between 2 points
distance_Class = []
```

```
for i in range(0,len(X)):
    dataPoint = X[i]
    disA = np.linalg.norm(dataPoint - centroids[0])
    disB = np.linalg.norm(dataPoint - centroids[1])
    disC = np.linalg.norm(dataPoint - centroids[2])
    distanceLS = [disA,disB,disC]
    minDis = min(distanceLS)
    if disA == minDis:
        #point closest to centroid 0
        distance_Class.append(cluster_label[0])
    elif disB == minDis:
        #point closest to centroid 1
        distance_Class.append(cluster_label[1])
    else:
        distance_Class.append(cluster_label[2])
data['DistanceClass'] = distance_Class
## Overall accuracy
accuracy = sum(data['L'] == data['DistanceClass'])/len(Y)
print("Overall accuracy is",accuracy)

print("Part 5")
##picking out result for label 1 and 2
new_df = data[data.L != 3]
accuracy = sum(new_df['L'] == new_df['DistanceClass'])/len(new_df.L)
print("Accuracy is:",accuracy)
## overvall confusion matrix
threeBythree = confusion_matrix(data.L.values,data.DistanceClass.values)
## getting rid of label 3
confusionMaxtrix = threeBythree[0:2,0:2]
tp = confusionMaxtrix[1,1]
fp = confusionMaxtrix[0,1]
tn = confusionMaxtrix[0,0]
fn = confusionMaxtrix[1,0]
print("tp: {0}, fp: {1}, tn: {2}, fn: {3}".format(tp, fp, tn, fn))
print("tpr: {0}, tnr: {1}".format(tp/(tp+fn),tn/(tn+fp)))
```

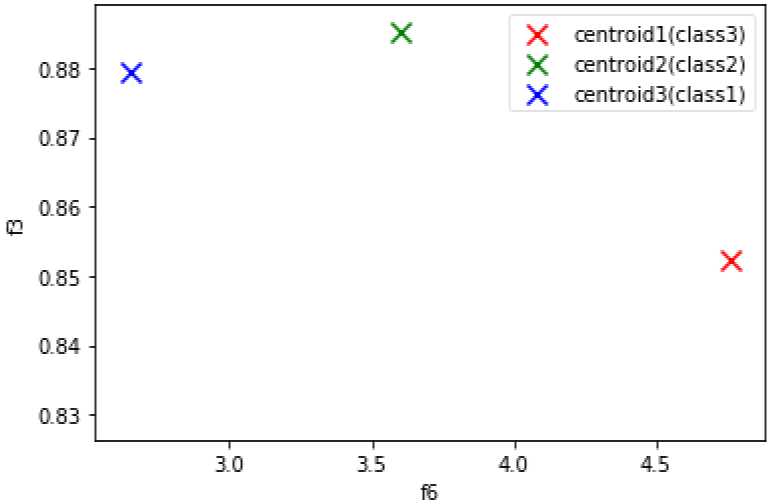
```
Part 1

No handles with labels found to put in legend.
```




```
Part 2
```




```
Part 3
Assign cluster 0 with class label 3
Centroid for cluster 0: [11.96441558 13.27480519  0.8522      5.22928571  2.8
7292208  4.75974026
  5.08851948]
Assign cluster 1 with class label 2
Centroid for cluster 1: [18.72180328 16.29737705  0.88508689  6.20893443  3.7
2267213  3.60359016
  6.06609836]
Assign cluster 2 with class label 1
Centroid for cluster 2: [14.63985915 14.45507042  0.87928169  5.56097183  3.2
7742254  2.65496056
  5.19192958]
```




```
Part 4
Overall accuracy is 0.8947368421052632
Part 5
Accuracy is: 0.8561151079136691
tp: 60, fp: 1, tn: 59, fn: 10
tpr: 0.8571428571428571, tnr: 0.9833333333333333
```