

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2020–2021 FALL SEMESTER**

**PROJECT-1 REPORT**

**(Arrays, Matrices, Methods, Random Numbers)**

**DELIVERY DATE**

--/--/---

**PREPARED BY**

05190000108, Hamit Can DAŞÇİ

05180000113, Simge Merve YAŞBAY

İçindekiler

[1) POINTS IN A 2D PLANE 2](#_Toc57390051)

[1.a Rastgele Nokta Üretimi 2](#_Toc57390052)

[1.a.1 Kodlar 2](#_Toc57390053)

[1.a.2 Ekran görüntüleri 3](#_Toc57390054)

[1.a.3 Açıklama 4](#_Toc57390055)

[1.b Uzaklık Matrisi 4](#_Toc57390056)

[1.b.1 Kodlar 4](#_Toc57390057)

[1.b.2 Ekran görüntüleri 5](#_Toc57390058)

[1.b.3 Açıklama 5](#_Toc57390059)

[2) CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN) ALGORITHM 6](#_Toc57390060)

[2.a KNN ile sınıflandırma 6](#_Toc57390061)

[2.a.1 Algoritma sözde-kodu 6](#_Toc57390062)

[2.b Banknot sınıflandırma 7](#_Toc57390063)

[2.b.1 Kodlar 7](#_Toc57390064)

[2.b.2 Ekran görüntüleri 8](#_Toc57390065)

[2.b.3 Açıklama 8](#_Toc57390066)

[2.c Başarı ölçümü 9](#_Toc57390067)

[2.c.1 Kodlar 9](#_Toc57390068)

[2.c.2 Ekran görüntüleri 10](#_Toc57390069)

[2.c.3 Açıklama 10](#_Toc57390070)

[2.d Listeleme 11](#_Toc57390071)

[2.d.1 Kodlar 11](#_Toc57390072)

[2.d.2 Ekran görüntüleri 12](#_Toc57390073)

[2.d.3 Açıklama 12](#_Toc57390074)

[Özdeğerlendirme Tablosu 12](#_Toc57390075)

# 1) POINTS IN A 2D PLANE

Visual Studio Code, 1.51, Python 3.9.1

## 1.a Rastgele Nokta Üretimi

### 1.a.1 Kodlar

Width = float(input("Width değeri : "))

Height = float(input("Height değeri : "))

Nokta\_sayisi = int(input("Oluşturulacak nokta sayısı: "))

def nokta\_uret(Nokta\_sayisi):

'''

nx2 lik boş matris oluşturuldu

random üretilen noktanın x ve y koordinatı üretilip geçici bir eleman adlı listede depolandı, böylelikle ileride x ve y değerlerine erişim daha kolaylaştırıldı.

eleman listesine koordinatlar x,y sırasıyla eklenildi ve sonrasında nx2 lik matrise eklendi.

'''

nx2 = []

for i in range(Nokta\_sayisi):

x\_point = uniform(0,Width)

y\_point = uniform(0,Height)

eleman = []

eleman.append(x\_point)

eleman.append(y\_point)

nx2.append(eleman)

return nx2

nx2 = nokta\_uret(Nokta\_sayisi) # İstenilen nokta sayısı kadar nokta 2D uzayda üretilip matrise atandı.

### 

### 1.a.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

----------

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1.a.3 Açıklama

## Matris ve liste(array) veri yapıları kullanıldı. 2 boyutlu düzlemde noktalar, girilen width ve height değerleri sınır alınarak rastgele olarak üretiliyor. Noktalar girdi olarak alınan n değişkeninin değeri tane üretilip nx2 boyutundaki matriste(iç içe liste kullanılarak oluşturuldu) tutuluyor. Nx2 matrisinin her bir satırı bir noktayı; her satırın ilk değeri x koordinatını, ikinci değeri y koordinatını saklıyor.

## 1.b Uzaklık Matrisi

### 1.b.1 Kodlar

def dist\_matrices(nx2):

nxn = [] # Distance matris oluşturuldu

for i in range(len(nx2)): # len fonksiyonu uzaklığı döndürüyor

nxn.append([]) # her ilk for döngüsünde nxn matris için satır oluşturuluyor.

for j in range(len(nx2)):

dist\_list = [] # uzaklıkların tutulduğu bir liste oluşturuluyor.

d = format(sqrt((nx2[i][0] - nx2[j][0])\*\*2 + (nx2[i][1] - nx2[j][1])\*\*2) , ".2f") # her bir noktanın diğer noktalara olan uzaklığı hesaplanıyor.

dist\_list.append(d)

nxn[i].append(dist\_list) # nxn matrisinin i. satırına uzaklıklar ekleniyor

return nxn

### 1.b.2 Ekran görüntüleri

### Width ve height değeri 100, n değeri 10 için program çıktısı

### metin içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.b.3 Açıklama

Matris ve liste(array) veri yapıları kullanıldı. Uzaklıkları tutmak için ayrı bir matris oluşturuldu, for döngüsünün başında her döngü için bir tane satır eklendi, noktalar arası uzaklık “iki nokta arası uzaklık” formulü ile hesaplanıp d değişkenine atandı. (matristeki her i=j için [i,j] noktasındaki uzaklığın sıfır çıkmasının sebebi spesifik bir noktanın tekrar kendisi ile uzaklığı 0 olması.) Daha sonra d değişkeni uzaklık matrisinin döngüdeki anlık satıra eklendi ve fonksiyon sonrası uzaklık matrisi return edildi.

# 

# 2) CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN) ALGORITHM

Visual Studio Code, 1.51, Python 3.9.1

## 2.a KNN ile sınıflandırma

### 2.a.1 Algoritma

'''

kodun try-catch bloğu içinde yazılmasının sebebi, ortaya çıkabilecek hataların konsolda yazdırılmasıdır.

‘’’

try:

def seperate\_rows(txt):

# verisetinin her bir satırını daha kolay kullanmak için ayrıştırıp data\_rows isimli listeye atadık.

Data\_rows = list()

for i in txt.readlines():

i = i.split(“ “)

‘’’

en sonda bulunan ‘\n’ kaçış dizisi dosyalarda yazılmadığı için veriler, ‘\n’ kadar alındı (satır sonuna kadar)

split metodu verilen parametrelere kadar objeyi elemanlarına ayırır ve her birini string türünde bir eleman olacak şekilde bir liste döndürür

‘’’

i[0] = i[0].replace(“\n” , “”) # Verilen verisetindeki her bir satırın sonunda alt satıra geçiren ‘\n’ kaçış dizisi bulunduğu için, değerleri alırken hata çıkmaması için kaldırıldı.

İ = i[0].split(“,”) # her bir satırdaki veriler “,” lerden ayırıp tekrar listeye ekledi

data\_rows.append(i) # veri setinin i. Satırı listeye atandı

return data\_rows

def dist\_list\_maker(para , data\_rows):

dist\_list = [] # İnput ile Baknot örneği olan para ile verisetindeki örnek paralarla aralarındaki uzaklığı tutacak liste.

For i in range(len(data\_rows)):

#uzaklığı verilen formüle göre hesaplıyoruz.

D = sqrt(

(float(data\_rows[i][0]) – para.varyans)\*\*2 +

(float(data\_rows[i][1]) – para.çarpıklık)\*\*2 +

(float(data\_rows[i][2]) – para.basıklık)\*\*2 +

(float(data\_rows[i][3]) – para.entropi)\*\*2

)

isReal = data\_rows[i][4] # verisetindeki örnek paraların türünü isReal değişkeninde depoluyoruz.

Temp\_tuple = how((d , isReal , i)) # uzaklığı , türünü , satır indexini geçici olarak bir demette depolayarak sonrasında uzaklık tutan dist\_list e ekliyoruz.

Dist\_list.append(temp\_tuple)

dist\_list.sort() # dist\_list listesini d ye göre artan biçimde sort metodu ile sıralıyoruz, tuttuğumuz index ile kaçıncı satırda olduğunu kaybetmiyoruz.

Return dist\_list

## 2.b Banknot sınıflandırma

### 2.b.1 Kodlar

def print\_classification(real , fake):

print(real , “ gerçek örnek, “ , fake , “ sahte örnek”, “\n”)

if int(real) > int(fake):

print(“Algoritmamızın tahminine göre banknot : Gerçek”, “\n”)

else:

print(“Algoritmamızın tahminine göre banknot : Sahte”, “\n”)

def print\_properties(dataRows , distList , k):

print(“Varyans”,”Çarpıklık”,”Basıklık”,”Entropi”,”Tür”,”Uzaklık” , sep=” “\*7)

print(“-“\*70)

for i in range(k):

index = distList[i][2] #index değişkeni, yakın örneğin veri setindeki satır değişkenini tutuyor

print(“-“ \* 70)

print(\*data\_rows[index] , sep=” “\*8,end=” “\*10) #en başa bırakılan yıldız liste elemanı yazdırırkenki elemanın parantezlerini ve virgüllerini kaldırıyor

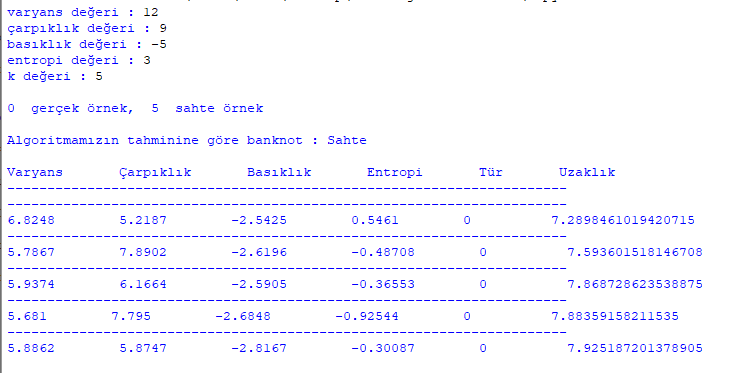
print(dist\_list[i][0])

### 

### 2.b.2 Ekran görüntüleri

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu



### 2.b.3 Açıklama

Liste(array) ve tuple veri yapısı kullanıldı . Geçici olarak paranın türünü tutması için isReal değişkenine en başta 1 değerini, sonradan sayaç değeri olarak kullanacağımız real ve fake değişkenlerine de 0 değerini atadık.

k\_list’ i , uzaklıkları artan olacak şekilde sıralanan ve kullanıcıdan alınan varyans, çarpıklık, basıklık ve entropi değerine göre hesaplanmış olan distList isimli 2 boyutlu bir listeden en yakın k adet komşusunu tutan bir değişken olarak belirledik. distList’ te her bir eleman, içinde uzaklık(d) , tür(isReal) ve veri setinde bulunduğu satır numarası (i) tutan bir demet olduğundan, tür bilgisini almak için k\_list değişkenindeki elemanın 1. Indeks değerine sahip olan değeri aldık ve bunun 1 mi 0 mı olduğunu if-else bloğu ile howl ettik.

K adet en yakın örneğin hepsinin türü kontrolden geçtikten sonra hangi değişkenin (real,fake) miktarı fazla ise paranın türünü (isReal) sayısı fazla olan değişken ile değiştirdik (eşit olması durumunda en yakın örneğin türünü kulandık).

## 2.c Başarı ölçümü

### 2.c.1 Kodlar

def success\_rate():

test\_k = int(input(“Test için k sayısı giriniz : “))

success\_guess = 0

rate = 0

test\_data = open(“test\_verileri.txt”)

big\_data = open(“test\_veri\_1172.txt”)

# yukarda tanımladığımız fonksiyonları başarı ölçümü kısmında tekrar kullandık

test\_data\_rows = seperate\_rows(test\_data)

big\_data\_rows = seperate\_rows(big\_data)

for i in range(len(test\_data\_rows)):

# data\_banknote\_authentication dosyasından alınan örnekleri Banknot örneği olarak oluşturuyor (karşılaştırma için)

test\_banknot = Banknot(

varyans= float(test\_data\_rows[i][0]),

çarpıklık= float(test\_data\_rows[i][1]),

basıklık= float(test\_data\_rows[i][2]),

entropi= float(test\_data\_rows[i][3]),

gerçek= float(test\_data\_rows[i][4])

)

test\_dist\_list = dist\_list\_maker(test\_banknot , big\_data\_rows) # test verileri için dist list oluşturuyor.

isReal = classification(test\_dist\_list , test\_k) # paranın türünü classification fonksiyonunu kullanarak tahmin edip isReal değişkenine atıyor

if (int(test\_banknot.gerçek) == int(isReal[0])): # tahmin ile gerçek türünü karşılaştırıyor. Classification sınıfı 3 adet değişkeni how(demet) içinde döndürdüğünden sadece tür bilgisini [0] ile alıyoruz

success\_guess += 1

def print\_test\_properties(test\_data\_rows, test\_dist\_list , test\_k):

print(“Varyans”,”Çarpıklık”,”Basıklık”,”Entropi”,”Tür”,”Uzaklık” , sep=” “\*7)

print(“-“\*70)

i = 0

while i != test\_k:

index = test\_dist\_list[i][2]

print(“-“ \* 70)

print(\*big\_data\_rows[index] , sep=” “\*8,end=” “\*10)

print(test\_dist\_list[i][0])

i += 1

print(“Bu paranın test sonucu : “ , “\n”, “Algoritma tahminine göre girilen banknotun türü : “ , isReal[0] , “ girilen banknotun asıl türü : “ , int(test\_banknot.gerçek))

print(“\*” \*100)

print\_test\_properties(test\_data\_rows , test\_dist\_list , test\_k)

rate = (success\_guess / len(test\_data\_rows)) \* 100

print(“Algoritmanın başarı oranı : “ , format(rate , “.2f”), “%”) #format ile düzgün yazdırılıyor

### 2.c.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

k=2 için konsol çıktısının bir kısmı

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

k=101 için Ekran çıktısının bir parçası

### 2.c.3 Açıklama

Liste (array) ve matris yapıları kullanıldı. Başarı ölçümünde kullanılan 200 adet test verisi ve geri kalan para örnekleri birbirinden ayrı 2 adet txt formatındaki dosyadan okundu. Her bir satırı bir adet para örneği saklayan matris yapısı, hem test verileri hem de geri kalan para örneklerini saklamak için 2 adet oluşturuldu. Test verilerindeki her bir örnek baştan başlanarak, özellikleri kullanılarak Banknot sınıfının bir örneği oluşturuldu. Paranın tahmini türü daha önce yazılan fonksiyonlar ile tahminlenip isReal adındaki değişkende saklandı. Bilinen gerçeklik değeri ile tahmin edilen gerçeklik değeri karşılaştırıldı, eğer doğru ise doğru tahmin sayısını tutan success\_guess değişkeninin değeri bir arttırıldı. Bu işlemler tüm test verilerine uygulandı. Başarılı tahmin sayısı ile kullanılan test verisi sayılarının yüzdesi hesaplanıp fonksiyon sonunda yazdırıldı.

## 2.d Listeleme

### 2.d.1 Kodlar

def print\_dataset():

how = input(“Bellekteki veri setini görmek istiyorsanız 1’i tuşlayın… “)

if how == “1”:

for i in data\_rows:

print(\*i)

### 2.d.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Konsol çıktısından bir parça

### 2.d.3 Açıklama

Fonksiyonda bir veri yapısı tanımlanmadı. Önceden oluşturulan data\_rows isimli verisetindeki satırları tutan liste yazdırıldı.

# Özdeğerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 15 | 15 |  |
| 1.b | 15 | 15 |  |
| Bölüm 1. Rapor | 10 | 10 |  |
| 2.a | 10 | 8 |  |
| 2.b | 10 | 10 |  |
| 2.c | 10 | 10 |  |
| 2.d | 10 | 10 |  |
| Bölüm 2. Rapor | 10 | 10 |  |
| Özdeğerlendirme Tablosu | 10 | 9 |  |

**Açıklama kısmında yapıldı, yapılmadı bilgisi veya hangi maddelerin nasıl yapıldığı kısaca yazılabilecektir.**