

**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**

14/12/2020

**PREPARED BY**

05190000031, Mert Ali Koçak

05190000043, Ahmet Hakan Demirel

05180000090, Savaş Soğancı

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 2](#_Toc57390054)

[1.a.3 Açıklama 2](#_Toc57390055)

[1.b Uzaklık Matrisi 2](#_Toc57390056)

[1.b.1 Kodlar 2](#_Toc57390057)

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

[1.b.3 Açıklama 2](#_Toc57390059)

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

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

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

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

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

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

[2.b.3 Açıklama 2](#_Toc57390066)

[2.c Başarı ölçümü 2](#_Toc57390067)

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

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

[2.c.3 Açıklama 3](#_Toc57390070)

[2.d Listeleme 3](#_Toc57390071)

[2.d.1 Kodlar 3](#_Toc57390072)

[2.d.2 Ekran görüntüleri 3](#_Toc57390073)

[2.d.3 Açıklama 3](#_Toc57390074)

[Özdeğerlendirme Tablosu 3](#_Toc57390075)

# 1) POINTS IN A 2D PLANE

//The platform, version, and programming language used

## 1.a Rastgele Nokta Üretimi

### 1.a.1 Kodlar

static double[,] matrixolustur(int widht, int height, int noktasayisi)

{

double[,] matrix = new double[noktasayisi, 2];

double nokta;

for (int i = 0; i < noktasayisi; i++)

{

for (int j = 0; j < 2; j++)

{

if (j == 0)

{

nokta = random.NextDouble() \* height;

nokta = Math.Round(nokta, 1);

matrix[i, j] = nokta;

}

else

{

nokta = random.NextDouble() \* widht;

nokta = Math.Round(nokta, 1);

matrix[i, j] = nokta;

}

}

}

return matrix;

static void matrixolusturYazdirma(double[,] matrix1)

{

int sayi = 1;

for (int i = 0; i < matrix1.GetLength(0); i++)

{

Console.WriteLine("{0}. noktanın x kordinatı: {1} {0}. noktanın y kordinatı: {2}", sayi, matrix1[i, 0], matrix1[i, 1]);

Console.WriteLine();

sayi++;

}

Console.WriteLine("-----------------------------------------------------------------------------");

}

### }1.a.2 Ekran görüntüleri

### metin içeren bir resim Açıklama otomatik olarak oluşturuldu

### 1.a.3 Açıklama

Genişlik, yüksekliği ve kaç nokta oluşturucağını parametre olarak alan metodumuz random.nextDouble kullarak 10 farklı x ve y değeri oluşturur. Bu noktaların değerleri 0-genişlik veya 0-yükseklik arasındadır. Daha sonrasında bu değerler noktalara atanır ve matrixolusturYazdirma metoduyla yazdırılır.

## 1.b Uzaklık Matrisi

### 1.b.1 Kodlar

static double[,] MatrixDm(double[,] matrix)

{

double[,] matrixDm = new double[matrix.GetLength(0), matrix.GetLength(0)];

for (int i = 0; i < matrixDm.GetLength(0); i++)

{

for (int j = 0; j < matrixDm.GetLength(0); j++)

{

matrixDm[i, j] = Math.Round(Math.Sqrt(Math.Pow(matrix[i, 0] - matrix[j, 0], 2) + Math.Pow(matrix[i, 1] - matrix[j, 1], 2)), 1);

}

}

return matrixDm;

}

static void matrixDmyazdirtablo(double[,] matrix2)//n>23 olursa konsol ekranında yer kalmayacağı için düzgün tablo düzgün olmayacaktır.

{

double[,] matrix = MatrixDm(matrix2);

Console.Write(" |");

for (int i = 0; i < matrix.GetLength(0); i++)

{

Console.Write("{0,-5} |", i + 1 + ".");

}

Console.WriteLine("");

Console.Write("------|");

for (int i = 0; i < matrix.GetLength(0) \* 7 - 1; i++)

{

Console.Write("-");

}

Console.WriteLine("|");

for (int i = 0; i < matrix.GetLength(0); i++)

{

Console.Write("{0,-5} |", i + 1 + ".");

for (int j = 0; j < matrix.GetLength(0); j++)

{

String virgulludeger = String.Format("{0:0.0}", matrix[i, j]);

Console.Write("{0,-5} |", virgulludeger);

}

Console.WriteLine("");

}

1.b.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.b.3 Açıklama

Oluşturduğumuz noktaların MatrixDM metoduyla aralarındaki uzaklıklarını hesaplıyoruz. Daha sonra matrixDMyazdirtablo metoduyla bunları tablo şeklinde yazdırıyoruz.

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

## 2.a KNN ile sınıflandırma

### 2.a.1 Algoritma

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Reflection;

namespace Projeodevi1

{

class paraVeri

{

public double varyans, carpiklik, basiklik, entropi, uzaklik;//Sınıfın değişkenlerini tanımladık.

public int sinif;

public paraVeri(double varyans, double carpiklik, double basiklik, double entropi, int sinif, double uzaklik)

{

this.varyans = varyans;

this.carpiklik = carpiklik;

this.basiklik = basiklik;

this.entropi = entropi;

this.sinif = sinif;

this.uzaklik = uzaklik;

}

public paraVeri()

{

}

}

class Program

{

static void Main(string[] args)

{

paraVeri[] veriseti1 = verisetiOlusturma("veriseti1.txt", 1372, 5);

paraVeri[] veriseti2 = verisetiOlusturma("veriseti2.txt", 200, 5);

paraVeri[] veriseti3 = verisetiOlusturma("veriseti3.txt", 1172, 5);

paraVeri girilenPara = paraBilgileri();

int kDeger = kDegeri();

knnhesapla(girilenPara, veriseti1, kDeger);

tabloyazdir(kDeger, girilenPara, veriseti1);

Console.WriteLine();

Console.WriteLine("200 verinin karşılaştırılması yapılacak.");

basariOrani(veriseti2, veriseti3);

Console.WriteLine("Listeleme işlemi için herhangi bir tuşa basın.");

Console.ReadLine();

listeleme(veriseti1);

Console.WriteLine("Çıkmak için herhangi bir tuşa basın.");

Console.ReadLine();

}

static int kDegeri()

{

int k;

Console.WriteLine("İstediğiniz K Değerini Giriniz: ");

k = Convert.ToInt32(Console.ReadLine());

return k;

}

static int knnhesapla(paraVeri para, paraVeri[] veritabani, int k)

{

for (int i = 0; i < veritabani.GetLength(0); i++)

{

double uzaklık = Math.Sqrt(Math.Pow(veritabani[i].varyans - para.varyans, 2) + Math.Pow(veritabani[i].carpiklik - para.carpiklik, 2) + Math.Pow(veritabani[i].basiklik - para.basiklik, 2) + Math.Pow(veritabani[i].entropi - para.entropi, 2));

veritabani[i].uzaklik = uzaklık;

}

var sonuc = veritabani.OrderBy(x => x.uzaklik);

paraVeri[] yakinlikdizisi = sonuc.ToArray();

int sinif0sayaci = 0;

int sinif1sayaci = 0;

for (int x = 0; x < k; x++)//k değeri kadar dönen döngü açıyoruz.

{

if (yakinlikdizisi[x].sinif == 1)

sinif1sayaci++;

else

sinif0sayaci++;

}

if (sinif0sayaci > sinif1sayaci)

return 0;

else if (sinif0sayaci < sinif1sayaci)

return 1;

else//

return yakinlikdizisi[0].sinif;

}

static void tabloyazdir(int k, paraVeri para, paraVeri[] veritabani)

{

int parasinif = knnhesapla(para, veritabani, k);

var sonuc = veritabani.OrderBy(x => x.uzaklik);

int sayac = 0;

Console.WriteLine("---------------------------------------------------------------------------------|");

Console.WriteLine("Varyans | Çarpıklık | Basıklık | Entropi | Sınıf | Uzaklık ");

Console.WriteLine("---------------------------------------------------------------------------------|");

foreach (paraVeri x in sonuc)

{

if (sayac < k)

{

Console.WriteLine(string.Format("{0,-10} | {1,-10} | {2,-10} | {3,-10} | {4,-10} | {5,-20}", x.varyans, x.carpiklik, x.basiklik, x.entropi, x.sinif, x.uzaklik));

sayac++;

}

else

{

Console.WriteLine("{0,-10} | {1,-10} | {2,-10} | {3,-10} | {4,-10} | {5,-20}", para.varyans, para.carpiklik, para.basiklik, para.entropi, parasinif, "Bizim Girdiğimiz Para");

break;

}

}//Paranın türünü yazdırıyoruz.

if (parasinif == 1)

Console.WriteLine("PARANIZ GERÇEK.");

else

Console.WriteLine("PARANIZ SAHTE");

Console.WriteLine();

}

static paraVeri paraBilgileri()

{

double varyans;

double basiklik;

double carpiklik;

double entropi;

Console.WriteLine("Paranızın Varyans Değerini Giriniz: ");

varyans = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

Console.WriteLine("Paranızın Çarpıklık Değerini Giriniz: ");

carpiklik = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

Console.WriteLine("Paranızın Basıklık Değerini Giriniz: ");

basiklik = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

Console.WriteLine("Paranızın Entropi Değerini Giriniz: ");

entropi = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

paraVeri girilenPara = new paraVeri();

girilenPara.varyans = varyans;

girilenPara.carpiklik = carpiklik;

girilenPara.basiklik = basiklik;

girilenPara.entropi = entropi;

return girilenPara;

}

static paraVeri[] verisetiOlusturma(string dosyakonumu, int satir, int sutun)//verilen dosyaların satır sütun sayısını bildiğimiz için direkt olarak parametrede yazabiliriz.

{

FileStream fs = new FileStream(dosyakonumu, FileMode.OpenOrCreate, FileAccess.Read, FileShare.ReadWrite);

StreamReader sr = new StreamReader(fs);

string[,] veriSeti = new string[satir, sutun];

paraVeri[] veriseti = new paraVeri[veriSeti.GetLength(0)];

int i = 0;

while (sr.EndOfStream != true)

{

string[] bolunmusSatir = sr.ReadLine().Split(',');

for (int j = 0; j < 5; j++)

{

veriSeti[i, j] = bolunmusSatir[j];

}

i++;

}

for (int j = 0; j < veriSeti.GetLength(0); j++)

{

veriseti[j] = new paraVeri();

veriseti[j].varyans = double.Parse(veriSeti[j, 0], CultureInfo.InvariantCulture);

veriseti[j].carpiklik = double.Parse(veriSeti[j, 1], CultureInfo.InvariantCulture);

veriseti[j].basiklik = double.Parse(veriSeti[j, 2], CultureInfo.InvariantCulture);

veriseti[j].entropi = double.Parse(veriSeti[j, 3], CultureInfo.InvariantCulture);

veriseti[j].sinif = Convert.ToInt32(veriSeti[j, 4]);

}

return veriseti;

}

static void basariOrani(paraVeri[] para, paraVeri[] veriseti)

{

int kDeger2 = kDegeri();

double dogruSayisi = 0;

for (int i = 0; i < para.GetLength(0); i++)

{

double tahmin = knnhesapla(para[i], veriseti, kDeger2);

tabloyazdir(kDeger2, para[i], veriseti);

if (tahmin == para[i].sinif)

{

dogruSayisi++;

}

}

double basariOrani = (dogruSayisi \* 100) / para.GetLength(0);

Console.WriteLine("Programın Başarı Oranı: %{0}", basariOrani);

}

static void listeleme(paraVeri[] veriseti)

{

for (int j = 0; j < veriseti.GetLength(0); j++)

{

Console.Write(veriseti[j].varyans);

Console.Write("," + veriseti[j].carpiklik);

Console.Write("," + veriseti[j].basiklik);

Console.Write("," + veriseti[j].entropi);

Console.Write("," + veriseti[j].sinif);

Console.WriteLine();

}

}

}

}

## 2.b Banknot sınıflandırma

### 2.b.1 Kodlar

static int knnhesapla(paraVeri para, paraVeri[] veritabani, int k)

{

for (int i = 0; i < veritabani.GetLength(0); i++)

{

double uzaklık = Math.Sqrt(Math.Pow(veritabani[i].varyans - para.varyans, 2) + Math.Pow(veritabani[i].carpiklik - para.carpiklik, 2) + Math.Pow(veritabani[i].basiklik - para.basiklik, 2) + Math.Pow(veritabani[i].entropi - para.entropi, 2));

veritabani[i].uzaklik = uzaklık;

}

var sonuc = veritabani.OrderBy(x => x.uzaklik);

paraVeri[] yakinlikdizisi = sonuc.ToArray();

int sinif0sayaci = 0;

int sinif1sayaci = 0;

for (int x = 0; x < k; x++)

{

if (yakinlikdizisi[x].sinif == 1

sinif1sayaci++;

else

sinif0sayaci++;

}

if (sinif0sayaci > sinif1sayaci)

return 0;

else if (sinif0sayaci < sinif1sayaci)

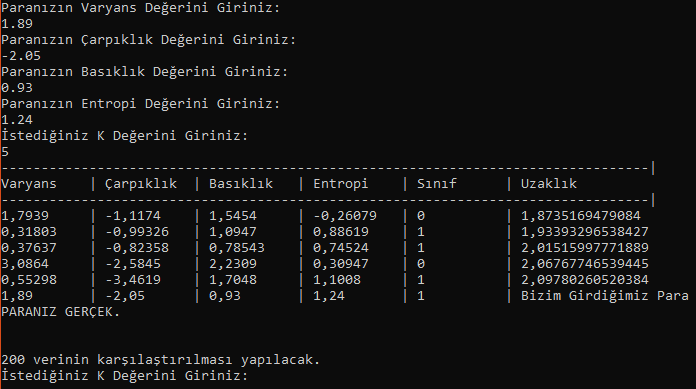
return 1;

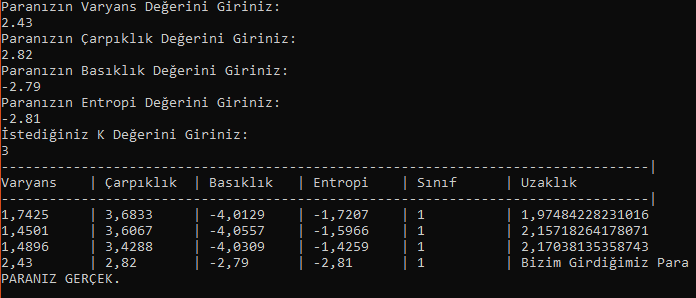
else

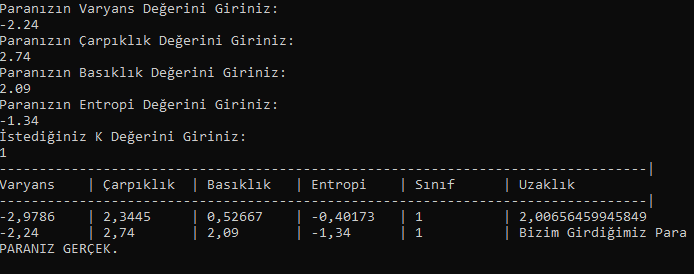
return yakinlikdizisi[0].sinif

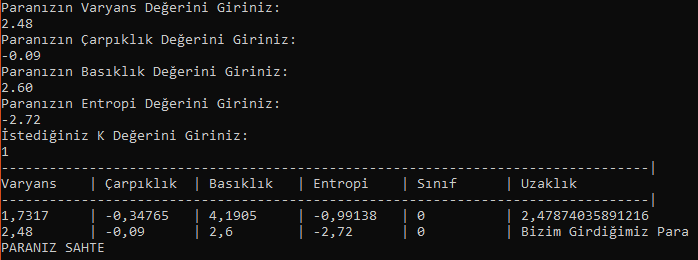
}

### 2.b.2 Ekran görüntüleri









### 2.b.3 Açıklama

Kullanıcının girdiği para verilerini ve istediği k adet nokta sayısını input olarak alıp verisetindeki en yakın k adet paranın veya paraların değerlerine olan uzaklığını hesaplayan ve buna göre paranın gerçek mi sahtemi olduğunu bulan program.

## 2.c Başarı ölçümü

### 2.c.1 Kodlar

static void basariOrani(paraVeri[] para, paraVeri[] veriseti)

{

int kDeger2 = kDegeri();//kNN için k değeri alıyoruz.

double dogruSayisi = 0

for (int i = 0; i < para.GetLength(0); i++)

{

double tahmin = knnhesapla(para[i], veriseti, kDeger2);

tabloyazdir(kDeger2, para[i], veriseti);

if (tahmin == para[i].sinif)

{

dogruSayisi++;

}

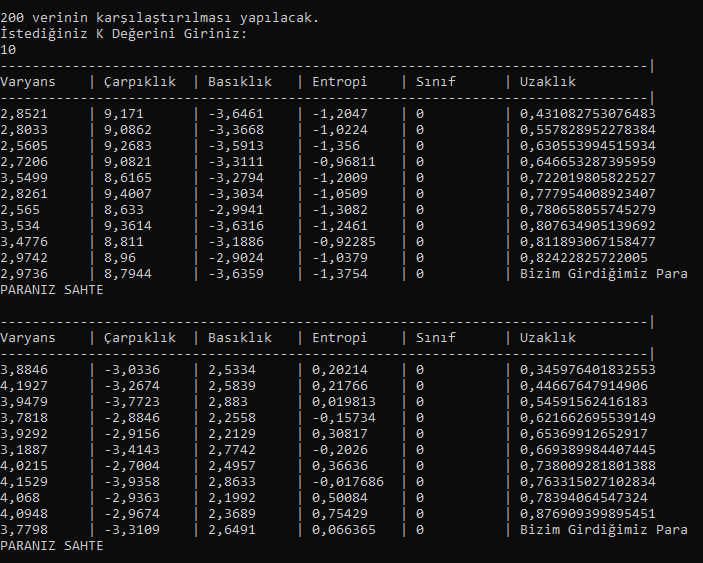
}

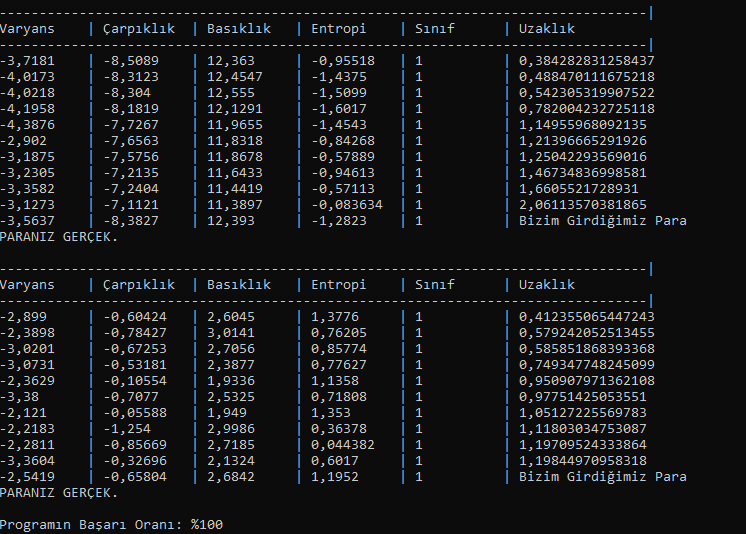
double basariOrani = (dogruSayisi \* 100) / para.GetLength(0);

Console.WriteLine("Programın Başarı Oranı: %{0}", basariOrani);

}

### 2.c.2 Ekran görüntüleri





### 2.c.3 Açıklama

Yazılan programın seçilen veriler ile test işlemi yapılıyor. Bİzim döndürdüğümüz sonuç ile paranın asıl gerçeklik değeri karşılaştırılıyor ve programın tutarlılığı hesaplanıp ekrana “Başarı Oranı” olarak bastırılıyor.

## 2.d Listeleme

### 2.d.1 Kodlar

static void listeleme(paraVeri[] veriseti)

{

for (int j = 0; j < veriseti.GetLength(0); j++)

{

Console.Write(veriseti[j].varyans);

Console.Write("," + veriseti[j].carpiklik);

Console.Write("," + veriseti[j].basiklik);

Console.Write("," + veriseti[j].entropi);

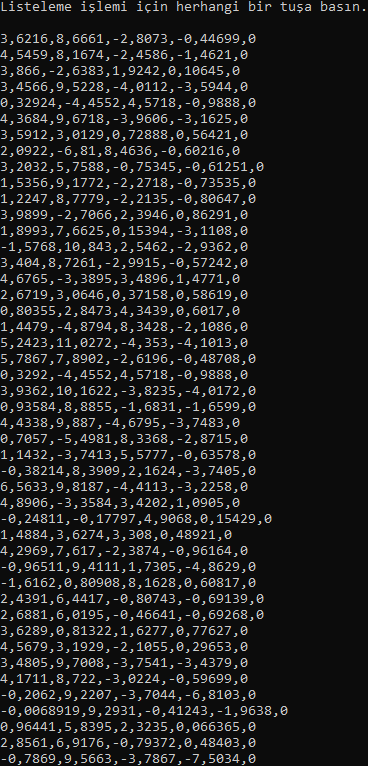
Console.Write("," + veriseti[j].sinif);

Console.WriteLine();

}

}

### 2.d.2 Ekran görüntüleri



### 2.d.3 Açıklama

Kullanılan verilerin ekrana satır satır bastırılöa işlemi.

# 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 | 10 |  |
| 2.b | 10 | 10 |  |
| 2.c | 10 | 10 |  |
| 2.d | 10 | 10 |  |
| Bölüm 2. Rapor | 10 | 10 |  |
| Özdeğerlendirme Tablosu | 10 | 5 |  |

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