

**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**

9/12/2020

**PREPARED BY**

05180000091, Kutay Avcı

05180000056, Emre Balkaya

İçindekiler

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

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

[1.a.1 Kodlar 3](#_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 3](#_Toc57390059)

[2) CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN) ALGORITHM 4](#_Toc57390060)

[2.a KNN ile sınıflandırma 5](#_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 12](#_Toc57390069)

[2.c.3 Açıklama 12](#_Toc57390070)

[2.d Listeleme 13](#_Toc57390071)

[2.d.1 Kodlar 13](#_Toc57390072)

[2.d.2 Ekran görüntüleri 13](#_Toc57390073)

[2.d.3 Açıklama 13](#_Toc57390074)

[Özdeğerlendirme Tablosu 14](#_Toc57390075)

# 1) POINTS IN A 2D PLANE

Visual Studio, 2019 16.8.0, C#

## 1.a Rastgele Nokta Üretimi

### 1.a.1 Kodlar

### public static System.Double[,] point(double width, double height, int n) //Rastgale Nokta Üretimi

### {

### Random random = new Random();

### double[,] nokta = new double[n, 2];

### for (int i = 0; i < n; i++)

### {

### double x = random.NextDouble() \* width; //Verilen yatay boyutta nokta üretimi (x)

### double y = random.NextDouble() \* height;//Verilen dikey boyutta nokta üretimi (y)

### nokta[i,0] = Math.Round(x,2);

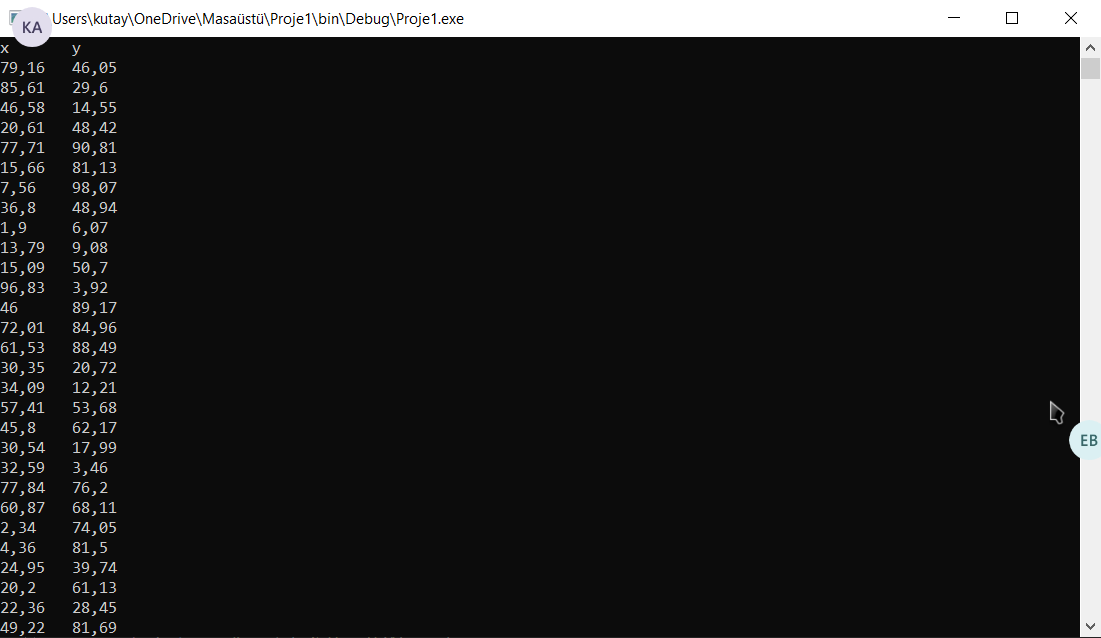
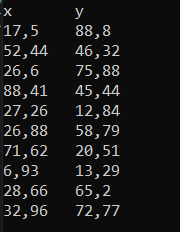
### nokta[i,1] = Math.Round(y,2);

### }

### return nokta;

### }

### 1.a.2 Ekran görüntüleri



### 1.a.3 Açıklama

İlk aşamada Random classını kullanarak oluşturacağımız noktanın x ve y değerlerini rastgele olarak aldık.Daha sonra for döngüsü ile oluşturduğumuz bu noktaları daha önce oluşturduğumuz nokta matrisinin içine attık. Yazdığımız fonksiyon da bu matrisi döndürüyor.

## 1.b Uzaklık Matrisi

### 1.b.1 Kodlar

### public static double uzaklık(double x1, double y1,double x2,double y2) //Euclidean distance func.

### {

### double uzaklık = Math.Sqrt(Math.Pow((x1 - x2), 2.0) + Math.Pow((y1 - y2), 2.0));

### //xler farkının karesi + yler farkının karesinin karekökü

### uzaklık = Math.Round(uzaklık, 2);

### return uzaklık;

### }

### public static System.Double[,] DM(double[,] nokta) // Uzaklık matrisinin oluşturulma fonksiyonu

### {

### double[,] dm = new double[nokta.GetLength(0), nokta.GetLength(0)];

### //Oluşturulan nokta sayısı boyutunda matris.

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

### {

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

### {

### double distance = uzaklık(nokta[i, 0], nokta[i, 1], nokta[j, 0], nokta[j, 1]);

### //Oluşturulan noktaların birbirine uzaklıklarının alınması

### dm[i, j] = distance;

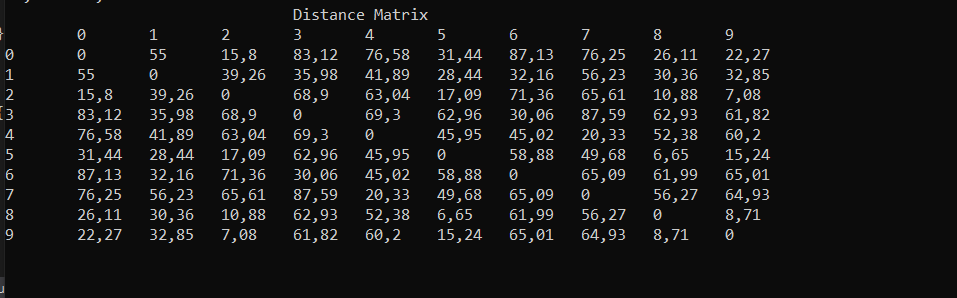
### }

### }

### return dm;

### }

### 1.b.2 Ekran görüntüleri



### 1.b.3 Açıklama

İlk fonksiyon uzaklık fonksiyonu öklid uzaklığına göre verilen iki noktanın birbirine uzaklığını hesaplayan ve döndüren bir fonksiyondur.

DM fonksiyonunda ise bir uzaklık matrisi oluşturduk.Önceden oluşturduğumuz noktaların birbirilerine uzaklığını uzaklık fonksiyonu ile alıp uzaklık matrisine attık ve bu matrisi döndürdük.

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

Visual Studio, 2019 16.8.0, C#

## 2.a KNN ile sınıflandırma

### 2.a.1 Algoritma

## 

## double[] uzaklık(test dizisi ,veri matrisi):

## double[] distlist;

## for (i = 0; i verilerin uzunluğuna kadar arttır):

## double distance = euclidian distance kullanarak hesapla

## distlist[i] = distance;

## return distlist;

## Sort(uzaklık dizisi, veri matrisi):

## while(bütün dizi sıralanana kadar):

## for (i = 0;uzaklık matrisinin uzunluğunun bir eksiğine kadar arttır):

## if (uzaklık matrisindeki elemanların ikişer karşılaştır büyükse 1 döndür):

## for (j = 0; 5 e kadar arttır):

## veriler matrisindeki büyük olan küçük olanla değiştir

## uzaklık matrisinindeki büyük olan elemanı küçük olanla değiştir

## int Classify(distance dizisi,veriler matrisi, int k):

## int tempbanknot;

## int falsecount = 0;

## int truecount = 0;

## for (int i = 0; i < k; i++):

## if (verinin sınıfı == 0):

## falsecount++;

## else:

## truecount++;

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

## Veriler matrisini bastır

## Uzaklık matrisini bastır

## if (doğru sınıf sayısı yanlış sınıftan fazlaysa):

## tempbanknot = 1;

## else if (yanlış sınıf sayısı doğru sınıftan fazlaysa):

## tempbanknot = 0;

## else:

## tempbanknot = uzaklığı en yakın olan verinin doğruluk sınıfını al

## return tempbanknot;

## 2.b Banknot sınıflandırma

### 2.b.1 Kodlar

### public static int Classify(double[] distance, double[,] veriler, int k)

### // Elde edilen sıralı uzaklığa ve girilen k değerine göre banknotun sınıflandırılması

### {

### int tempbanknot;

### int falsecount = 0;

### int truecount = 0;

### Console.WriteLine("Varyans\t\tÇarpıklık\tBasıklık\tEntropi\t\tSınıf\t\tUzaklık");

### for (int i = 0; i < k; i++)

### {

### if (veriler[i, 4] == 0)

### //verisetinde en yakın k verinin doğruluk değerlerine göre yeni banknotun doğruluk değeri tahminlenir.

### falsecount++;

### else

### truecount++;

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

### {

### Console.Write(veriler[i, j] + "\t\t");

### }

### Console.WriteLine(distance[i]);

### }

### if (truecount > falsecount)

### {

### tempbanknot = 1;

### }

### else if (falsecount > truecount)

### tempbanknot = 0;

### else

### {

### tempbanknot = (int)veriler[0, 4];

### }

### if (tempbanknot == 1)

### Console.WriteLine("Özellikleri verilen banknot gerçektir");

### else

### Console.WriteLine("Özellikleri verilen banknot sahtedir");

### return tempbanknot;

### }

### 2.b.2 Ekran görüntüleri

### 2.b.3 Açıklama

Girilen k değerine göre sıralanmış verilerden en yakın k tanesi alınır ve ekrana bastırılır.Alınan bu verilerin doğruluk sınıfları sayılır.En çok olan sınıf yeni banknotun doğruluk sınıfı olarak tahminlenir.Eğer sınıf sayıları eşitse en yakın olan verinin sınıfı olarak tahminlenir.

## 2.c Başarı ölçümü

### 2.c.1 Kodlar

### double[,] testverileri = new double[200, 5];//verisetinden doğruluk değeri 0 olan son 100 veriyi ve 1 olan son 100 veriyi tutmak için dizi açtık.

### for (int j = 662; j < 762; j++)// doğruluk değeri 0 olan son 100 verinin testverileri dizisine aktarımı.

### {

### for (int k = 0; k < 5; k++)

### {

### testverileri[j - 662, k] = verilerbaşarı[j, k];

### }

### }

### for (int j = 1272; j < 1372; j++)//doğruluk değeri 1 olan son 100 verinin testverileri dizisine aktarımı.

### {

### for (int k = 0; k < 5; k++)

### {

### testverileri[j - 1172, k] = verilerbaşarı[j, k];

### }

### }

### double[,] verilerbaşarıson = new double[1172, 5];// verisetinden aldığımız 200 verinin eksiltilmiş hali ile verisetinin yeni bir diziye aktarılması.

### for (int j = 0; j < 662; j++)

### {

### for (int k = 0; k < 5; k++)

### {

### verilerbaşarıson[j, k] = verilerbaşarı[j, k];

### }

### }

### for (int j = 762; j < 1272; j++)

### {

### for (int k = 0; k < 5; k++)

### {

### verilerbaşarıson[j - 100, k] = verilerbaşarı[j, k];

### }

### }

### double[] sınıfbaşarı = new double[testverileri.GetLength(0)];// verisetinden aldığımız 200 verinin tahminlenen doğruluk değerlerini içeren bir dizi

### int a;

### while (true) //Hata kontrolü

### {

### try

### {

### Console.WriteLine("Başarı oranı için K değerini giriniz:");

### a = Convert.ToInt32(Console.ReadLine());

### break;

### }

### catch (System.FormatException e)

### {

### Console.WriteLine("Hatalı değer girdiniz.Lütfen sayı giriniz!");

### }

### }

### Console.WriteLine("Veriler yükleniyor lütfen bekleyiniz...");

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

### {

### double[] testtemp = new double[5];

### for (int k = 0; k < 5; k++)

### {

### testtemp[k] = testverileri[j, k];//testverileri dizisindeki her eleman veriseti ile karşılaştırılıp sınıflandırılır

### }

### double[] uzaklıkbaşarı = distance(testtemp, verilerbaşarıson);

### Sort(uzaklıkbaşarı, verilerbaşarıson);

### int başarıclassify = ClassifyBaşarı(uzaklıkbaşarı, verilerbaşarıson, a);

### sınıfbaşarı[j] = Convert.ToDouble(başarıclassify);

### }

### double[] gerçeksınıf = new double[testverileri.GetLength(0)];// burada ise verisetinden alınan 200 verinin gerçek doğruluk değerleri tutulur.

### for (int k = 0; k < gerçeksınıf.GetLength(0); k++)

### gerçeksınıf[k] = testverileri[k, 4];

### Console.WriteLine("Gerçek sınıf" + "\t\t" + "Tahminlenen sınıf");

### for (int k = 0; k < gerçeksınıf.GetLength(0); k++)

### {

### Console.WriteLine(gerçeksınıf[k] + "\t\t" + sınıfbaşarı[k]);// tahminlenen doğruluk verileriyle gerçek doğruluk verileri bastırılır.

### }

### double count = 0.0;

### for (int k = 0; k < gerçeksınıf.GetLength(0); k++)// tahminlenen doğruluk verilerinin kaç tanesinin gerçek verilerle uyuştuğu ölçülür.

### {

### if (gerçeksınıf[k] == sınıfbaşarı[k])

### {

### count++;

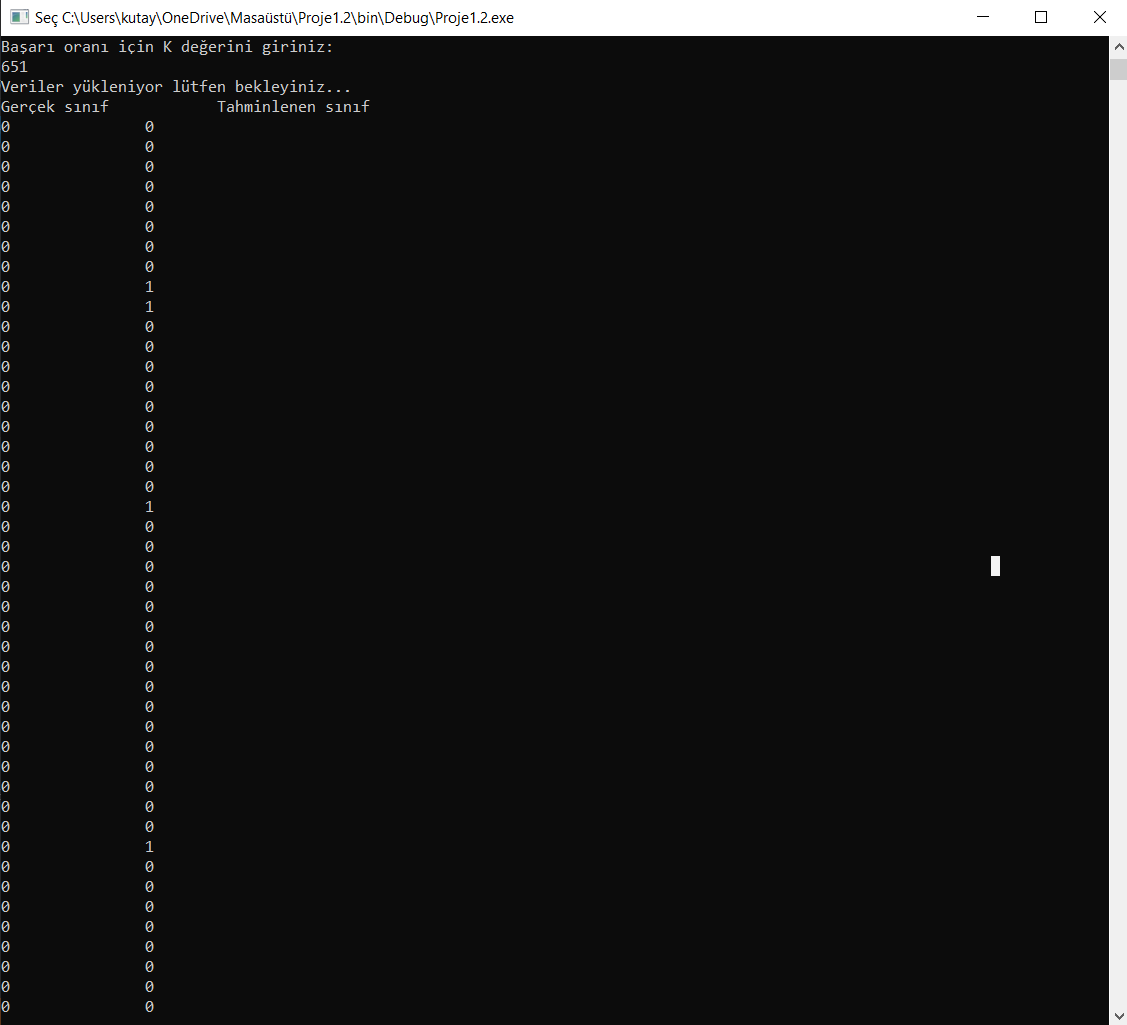
### }

### }

### double başarıoranı = count / gerçeksınıf.GetLength(0) \* 100;// ölçülen uyuşma sayısı ile programın başarı oranı hesaplanır.

### Console.WriteLine("Başarı oranı: %" + başarıoranı);

### 2.c.2 Ekran görüntüleri





### 2.c.3 Açıklama

İstenilen 200 tane veriyi verilen veri listesinden çıkararak bir matrisi attık.Daha sonra bu 200 verinin çıkartılmış halini almak için başka bir veri matrisi oluşturup kalan verileri de buraya attık.Daha sonra bu 200 verinin knn algoritmasını kullanarak kalan 1172 veriye göre doğruluk sınıfını tahminleyip elimizdeki gerçek sınıfları ile karşılaştırıp oranlayarak başarı oranını elde ettik.

## 2.d Listeleme

### 2.d.1 Kodlar

### Console.WriteLine("Bellekteki Veriseti: ");

### Console.WriteLine("Varyans\t\tÇarpıklık\tBasıklık\tEntropi\t\tSınıf");

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

### // Verisetinin ilk ve değiştirilmemiş hali yazdırılır.

### {

### for (int y = 0; y < 5; y++)

### {

### Console.Write(verilerdegismeyen[j, y] + "\t\t");

### }

### Console.WriteLine();

### }

### 2.d.2 Ekran görüntüleri

### 2.d.3 Açıklama

Verisetinin ilk ve değiştirilmemiş halini for döngüsü ile yazdırdık.

# Özdeğerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 15 | 15 | Yapıldı |
| 1.b | 15 | 15 | Yapıldı |
| Bölüm 1. Rapor | 10 | 10 | Yapıldı |
| 2.a | 10 | 10 | Yapıldı |
| 2.b | 10 | 10 | Yapıldı |
| 2.c | 10 | 10 | Yapıldı |
| 2.d | 10 | 10 | Yapıldı |
| Bölüm 2. Rapor | 10 | 7 | Yapıldı |
| Özdeğerlendirme Tablosu | 10 | 8 | Yapıldı |

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