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

13/12/2020

PREPARED BY

05190000061, Oktay Kaloğlu

İçindekiler

1) POINTS IN A 2D PLANE

//Visual Studio Community 2019, 16.8.1 and C# used to develop this program.

1.a Rastgele Nokta Üretimi

1.a.1 Kodlar

```
class Points
{
    class Point
    {
        public double x = 0.0;
        public double y = 0.0;
        public Point(double x1 , double y1)
        {
            this.x = x1;
            this.y = y1;
        }
    }
    private int n = 0;
    private int width=0;
    private int height= 0;
    private Point[] points ;
    private double[,] theMatrix;
    public Points(int num , int wid, int hei){

        this.n = num;
        this.width = wid;
        this.height = hei;
        points = fillWithPoints();
        outPutPoints();
        theMatrix = distanceMatrix();
        outPutDistanceMatrix();
    }

    private Point[] fillWithPoints()
    {
        Point[] tempPoints = new Point[n];
        for (int i = 0; i < n; i++)
        {
            Random random = new Random();
            double x1 = Convert.ToDouble(random.Next(0, height * 10))/10;
            double y1 = Convert.ToDouble(random.Next(0, width * 10))/10;
            Point pointi = new Point(x1,y1) ;
            tempPoints[i] = pointi;
        }
        return tempPoints;
    }

    private void outPutPoints() {
        for (int i = 0; i < n; i++)
        {
            Console.Write("point " + (i+1) + ": (");
            Console.Write(points[i].x);
            Console.Write(",");
            Console.Write(points[i].y);
            Console.Write(")");
            Console.WriteLine("");
        }
        Console.WriteLine();
        Console.WriteLine();
    }
```

1.a.2 Ekran görüntüleri

```
point 1: (35.1,2.5)
point 2: (81.6,96.4)
point 3: (97.3,99.2)
point 4: (59.4,45.5)
point 5: (23.3,95.4)
```

```
point 1: (58.7,43.9)
point 2: (11.9,38.6)
point 3: (37.3,70.4)
point 4: (62.1,77.9)
point 5: (62.3,23.2)
point 6: (58.8,91.1)
point 7: (37.3,33.5)
point 8: (32,2.1)
point 9: (16.4,40.6)
point 10: (76.9,21.2)
```

1.a.3 Açıklama

Algoritma geliştirilirken dizi(double[],Point[]) ve noktanın x ve y değerlerini tutması için Point adlı nesne kullanılmıştır.

Program rastgele nokta üretimini ve uzunluk matrisini sırası ile points adlı nesnenin constructoru içerisinde gerçekleştirmektedir.

Algortima giriş verileri olan genişlik veya yükseklik değerlerine göre rastgele değerler üretililir. x ve y double verilerini n tane nokta için ayrı ayrı üretmektedir. Üretilen noktalar genişliğin 10 katı ile RandİnT metodu ile rastgele üretlir ve sonra 10 a bölünmesiyle 0.0 formatında double a dönüştürülür. Rand double yerine rand int seçilmesinin nedeni gereksiz yere noktadan sonra sadece bir basamağın yeterli olmasıdır.

1.b Uzaklık Matrisi

1.b.1 Kodlar

//bu kod Points adlı classın üyesidir

```
public double[,] distanceMatrix()
{
    double[,] disMat = new double[n,n];
    double dist = 0.0;
    for(int i = 0; i<n; i++)
    {
       for(int j =0; j <= i; j++)
        {

            dist = distanceCalculator(points[i], points[j]);
            disMat[i,j]=dist;
            disMat[j, i] = dist;
        }
    }
    double distanceCalculator(Point p1, Point p2) {
        return Math.Sqrt( Math.Pow((p1.x-p2.x),2)+ Math.Pow((p1.y - p2.y), 2)) ;
    }
    return disMat;
}
```

```
        public void outPutDistanceMatrix()
        {
            string spc = new string(' ', 6*(n / 2));
            Console.WriteLine(string.Format(spc+"Distance Matrix"));
            Console.Write("      ");
            for (int i=0;i<n;i++)
            {
                Console.Write(string.Format("{0,6:0.0}", "n" + i, "  "));
            }
            Console.WriteLine();

            for (int i = 0; i < n; i++)
            {
                Console.Write(string.Format("{0,6:0.0}", "n"+i, "  "));

                for (int j = 0; j < n; j++)
                {
                    Console.Write(string.Format("{0,6:0.0}",  theMatrix[i, j] ,"  "));

                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }

    }
```

1.b.2 Ekran görüntüleri

```
          Distance Matrix
        n0    n1    n2    n3    n4
n0   0.0 104.8 115.0  49.4  93.6
n1 104.8   0.0  15.9  55.5  58.3
n2 115.0  15.9   0.0  65.7  74.1
n3  49.4  55.5  65.7   0.0  61.6
n4  93.6  58.3  74.1  61.6   0.0
```

```
                              Distance Matrix
        n0    n1    n2    n3    n4    n5    n6    n7    n8    n9
n0   0.0  47.1  34.1  34.2  21.0  47.2  23.8  49.6  42.4  29.1
n1  47.1   0.0  40.7  63.8  52.7  70.4  25.9  41.7   4.9  67.3
n2  34.1  40.7   0.0  25.9  53.4  29.8  36.9  68.5  36.4  63.2
n3  34.2  63.8  25.9   0.0  54.7  13.6  50.9  81.6  59.0  58.6
n4  21.0  52.7  53.4  54.7   0.0  68.0  27.0  36.9  49.1  14.7
n5  47.2  70.4  29.8  13.6  68.0   0.0  61.5  92.9  65.9  72.2
n6  23.8  25.9  36.9  50.9  27.0  61.5   0.0  31.8  22.1  41.5
n7  49.6  41.7  68.5  81.6  36.9  92.9  31.8   0.0  41.5  48.8
n8  42.4   4.9  36.4  59.0  49.1  65.9  22.1  41.5   0.0  63.5
n9  29.1  67.3  63.2  58.6  14.7  72.2  41.5  48.8  63.5   0.0
```

1.b.3 Açıklama

Algoritma geliştirilirken dizi(double[],Point[]), iki boyutlu dizi(double[,]) ve noktanın x ve y değerlerini tutması için Point adlı nesne kullanılmıştır.

Algoritma points adlı dizinin noktalararının birbirlerine uzaklıklarının nXn lik bir matrise kaydedilmesi ve bu matrisin çıktısını vermektedir. Iki nokta arasındaki uzaklık farkı bulunurken Math metotları kullanılmıştır.

Matris oluşturulurken n1 in n2 ye olan uzaklığı ile n2 nin n1 e olan uzaklığı aynı olduğu için bu verinin bir daha hesaplanmasına gerek kalmamaktadır. Her nokta for içerisinde işleme alınırken yanlızca kendi sırası ve sırasından önceki noktaların uzaklıklarının hesaplanmasıyla bütün matris doldurulabilmiştir.(tablo 1 deki alt üçgen gibi)

her n nokta için n*n tane uzaklık hesaplaması yapılması yerine uzaklık hesaplaması yalnızca (1+2+3+4.....n) yani (n*(n+1)/2) kere yapılarak gereksiz işlem gücü tüketiminden kaçınılmıştır.

Distance Matrix

	0	1	2	3	4	5
0	0.0	1.2	0.5	4.7	5.6	4.9
1	1.2	0.0	3.1	2.0	1.4	4.0
2	0.5	3.1	0.0	6.1	2.8	1.9
3	4.7	2.0	6.1	0.0	2.1	3.5
4	5.6	1.4	2.8	2.1	0.0	3.3
5	4.9	4.0	1.9	3.5	3.3	0.0

tablo 1

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

//Visual Studio Community 2019, 16.8.1 and C# used to develop this program.

2.a KNN ile sınıflandırma

2.a.1 Algoritma

//kendi geliştridiğim algoritma birden fazla dış metot yardımı ile çalıştığı için burda açıklama amaçlı sözlü bir kodu buraya yazdım.

```
static List<Data> KNNAlgo(Data dat, int k, List<Data> set)
{
        List<Data> kAdetYakınData =new List<Data>();
        for (int i = 0; i < set.Count; i++)
        {
                set[i].distance = Math.Sqrt(Math.Pow(set[i].varyans - dat.varyans, 2) +
                Math.Pow(set[i].carpıklık - dat.carpıklık, 2) + Math.Pow(set[i].basıklık -
                dat.basıklık, 2) + Math.Pow(set[i].entropi - dat.entropi, 2));
        }
        set.Sort();//her hangibir algoritma ile uzaklığa göre sıralama yapılmalıdır. Ben kendi kodumda
                //linkedlist kullandım.

        int toplamlar=0;
        For(int i = 0; i<k; i ++ ){
                kAdetYakınData.add(set[i]);
                sum+=set[i].tur;
        }

        if (sum / k > 0.5)//kendisine yakın paralarda gerçek sayısı fazla ise
       {
        d.knnTur = 1;
       }else if (sum / k < 0.5)
       {
        d.knnTur = 0;
       }
        Else
        {
        d.knnTur = kAdetYakınData[0].tur;
// eğer k değerine göre iki tür banknot sayısı eşitse en yakın paranın türüne eşittir.
```

```
        }
        return kAdetYakınData;
}
```

2.b Banknot sınıflandırma

2.b.1 Kodlar

```
static void bankotSiniflandirma(List<Data> set)
        {
            String[] degerler = { "varyans", "çarpıklık", "basıklık", "entropi" };
            while (true)
            {
                double[] data = new double[4];
                List<Data> rList;
                Console.WriteLine("Lütfen bir k değeri giriniz: ");
                int k = Convert.ToInt32(Console.ReadLine());
                for (int i = 0; i < 4; i++)
                {
                    Console.WriteLine("Lütfen bir" + degerler[i] + "değeri giriniz: ");
                    data[i] = Convert.ToDouble(Console.ReadLine());
                }

                Data dat = new Data(0, data[0], data[1], data[2], data[3], -1, -1);//turu bilinmediği
için tur yerine -1 girdim

                rList = KNNAlgo(dat, k, set);
                Console.Out.WriteLine("knn ile hesaplanan banknot türü: " + dat.knnTur);
                dataListOut(rList);

                Console.WriteLine("başka bir banknot için de işlem yapmak ister misiniz ?(e/E - h/H):
");
                string durum = Console.ReadLine();
                if (durum == "h" || durum == "H")
                {
                    break;
                }

            }
        }

static void dataListOut(List<Data> set,int k =0)//List<Data> türünden listenin yazdırılması

        {
            bool durum = k != 2;// en son verilerin tamamı yazdırılırken hesaplanmış uzaklıkların
yazdırılmasına gerek yoktur.
            Console.Write(string.Format("{0,20:0.0}", "Varyans"));
            Console.Write(string.Format("{0,20:0.0}", "Çarpıklık"));
            Console.Write(string.Format("{0,20:0.0}", "Basıklık"));
            Console.Write(string.Format("{0,20:0.0}", "Entropi"));
            Console.Write(string.Format("{0,20:0.0}", "Tür"));
            if (durum)
            {
                Console.Write(string.Format("{0,20:0.0}", "Uzaklık"));
            }
            Console.Out.WriteLine("");
            for (int i = 0; i < set.Count; i++)
            {
                Console.Write(string.Format("{0,20}", set[i].varyans));
                Console.Write(string.Format("{0,20}", set[i].carpıklık));
                Console.Write(string.Format("{0,20}", set[i].basıklık));
                Console.Write(string.Format("{0,20}", set[i].entropi));
                Console.Write(string.Format("{0,20}", set[i].tur));
                if (durum) {
                    Console.Write(string.Format("{0,20:0.00000000}", set[i].distance));
                }
                Console.Out.WriteLine("");
            }
        static List<Data> KNNAlgo(Data dat, int k, List<Data> set)
        {
```

```
            MyList linkedList = new MyList() ;
            for (int i = 0; i < set.Count; i++)
            {
                set[i].distance = Math.Sqrt(Math.Pow(set[i].varyans - dat.varyans, 2) +
Math.Pow(set[i].carpıklık - dat.carpıklık, 2) + Math.Pow(set[i].basıklık - dat.basıklık, 2) +
Math.Pow(set[i].entropi - dat.entropi, 2));
                linkedList.add(set[i]);//sıralanarak ekleniyor araya item ekleneceği zaman bütün
itemlerin indexlerinin arttırılmasına gerek kalmıyor
            }
            return linkedList.returnList(k,dat);// girilen datanın knn ile türünün bulunması ve
Linkedlist yerine işlem kolaylığı için List<data> sınıfından bir liste döndürüyor


        }

  class MyList{
            public Node head;
            public int count=0;
            internal class Node
            {
                internal Data dat;
                internal Node prev;
                internal Node next;
                public Node(Data d)
                {
                    this.dat = d;
                    this.prev = null;
                    this.next = null;
                }

            }

            public void add(Data d)
            {
                if (d!=null) {
                    Node tempNode = head;
                    Node data = new Node(d);
                    if (head != null)
                    {

                        while(true)
                        {
                            if (tempNode.dat.distance < d.distance)
                            {
                                if (tempNode.next != null)
                                {
                                    tempNode = tempNode.next;//listenin dönülmesi
                                } else//adding to end of the list
                                {
                                    count++;
                                    tempNode.next = data;
                                    data.prev = tempNode;
                                    break;
                                }
                            }
                            else//doğru yer bulundu ekleme yapılması lazım
                            {
                                if (tempNode != head)//ilk elemandan sonra eklenme
                                {
                                    Node current = tempNode;
                                    current = current.prev;
                                    data.prev = current;
                                    data.next = tempNode;
                                    current.next= data;
                                    tempNode.prev = data;
                                    break;
                                }
                                else//ilk elemanın önüne eklmek için
                                {
                                    tempNode.prev = data;
                                    data.next = tempNode;
                                    head = data;
                                    count++;
                                    break;
```

```
                        }
                    }
                }

            }
            else//liste boşsa
            {
                head = data;
                count++;
            }
        }
       }

        public List<Data> returnList(int k,Data d)//listenin baştan k ya kadar olanının geri
döndürülmesi ve giriş datasının boğruluk değernin ortaya çıkartılması
        {
            List<Data> forReturn = new List<Data>(k);
            Node tempNode = head;
            double sum = 0;
            for(int i = 0; i < k && tempNode !=null; i++)
            {
                sum +=tempNode.dat.tur;
                forReturn.Add(tempNode.dat);
                tempNode=tempNode.next;
            };
            if (sum / k > 0.5)//eğer kendisine yakın değerlerde doğru sayısı fazla ise
            {
                d.knnTur = 1;
            }else if (sum / k < 0.5)
            {
                d.knnTur = 0;
            }
            else// eğer sahte ve doğru tür sayısı eşitse baştaki elemanın değerindedir.
            {
                d.knnTur = head.dat.tur;
            }
            return forReturn;
        }

       public void lListOut()//kontrol amaçlı link listin itemlerinin yazdırılması için
        {
            Node temp = head;
            while(temp!=null)
            {
                Console.Out.WriteLine(temp.dat.toString());
                temp = temp.next;
            }
        }



    }
```

2.b.2 Ekran görüntüleri

```
Lütfen bir k değeri giriniz:
5
Lütfen birvaryansdeğeri giriniz:
0
Lütfen birçarpıklıkdeğeri giriniz:
0
Lütfen birbasıklıkdeğeri giriniz:
0
Lütfen birentropideğeri giriniz:
0
knn ile hesaplanan banknot türü: 1
          Varyans          Çarpıklık          Basıklık           Entropi               Tür            Uzaklık
       -0.0012852            0.13863           -0.19651         0.0081754                 1         0.24063031
         -0.40951           -0.15521           0.060545          -0.088807                 1         0.45093344
          -0.4294           -0.14693           0.044265           -0.15605                 1         0.48195827
           0.3434            0.12415           -0.28733            0.14654                 1         0.48720559
        -0.092194            0.39315           -0.32846           -0.13794                 1         0.53849798
başka bir banknot için de işlem yapmak ister misiniz ?(e/E - h/H):
```

2.b.3 Açıklama

//Kullanılan veri yapıları ve algoritmanın kısaca anlatımını burada gerçekleştiriniz

Algoritma geliştirilirken dizi(double[]), nesne listesi(List<Data>) kendim için geliştirdiğim Mylist adlı çift bağlı liste kullanılmıştır.

Algoritmadaki asıl amaç giriş banknotun verilerinin veri dosyasındaki verilerle karşılaştırılıp aralarındaki farkın bulunarak giriş yapılmış verinin gerçek mi sahte mi olduğunun tahmin edilmesi.

Algoritma öncelikle kullanıcı girişi almaktadır. Bu alınan veriler bir Data nesnesine donüştürülerek KNNALgo metotuna bütün veriler ile verilir. Girşi alınan veri nesne listesindeki her bir veri ile tek tek karşılaştırılarak bulunan uzunluklarla birlikte veri nesneleri çift bağlı listeye eklenmektedir. İlerde tekrardan bütün listenin baştan sona balon sıralaması gibi algoritmalarla tekrar sıralanması işlem gücünün boşa sarf edeceği için normal listeler tercih edilmemiştir. Çift bağlı listede ekleme işlemi, eklenicek yerin bulunması ve gerekli referans noktalarının değiştirilmesi ile kolayca ve fazla işlem gücü gerekmeden halledebilmektedir. Normal bir listede araya ekleme işlemi yapılsaydı eklenecek yerden sonraki bütün nesnelerin indexlerinin tek tek bir arttırılması gerekcekti.

Çift bağlı listede bütün eklemeler bittikten sonra gereken k kadar yakın veriler bir nesne listesine dönüştürülür ve bu dönüştürme sırasında doğruluk değerleri toplanır.Toplanan doğruluk değerlerinin k ya bölünmesi ile giriş yapılan veriye knnTur verisine tahminlenen doğruluk değeri eklenir ve döndürülen listedeki verilerin uzaklık ve diğer değerlerinin çıktıları alınır.

H veya h girişi yapılana kadar bu veri setinde başka k ve banknot değerleri içinde bu hesaplamaların hepsi tekrar tekrar yapılmaktadır.

2.c Başarı ölçümü

2.c.1 Kodlar

```
static void basariOlcumu(List<Data> testData, List<Data> dataSet)
        {

            while (true)
            {
                Console.WriteLine("Lütfen bir k değeri giriniz: ");
                int k = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("K " + k + " için hesaplanıyor");
                for (int i = 0; i < testData.Count; i++)
                {
                    List<Data> rList = KNNAlgo(testData[i], k, dataSet);
                    Console.WriteLine("Veri " + i+" asıl türü: "+testData[i].tur+" knn ile hesaplanan
türü: "+testData[i].knnTur);
                    dataListOut(rList);
                    Console.WriteLine();
                }

                //knn ile tahmin    gerçek tür
                double dogruSay= 0;

                for(int i = 0; i < 200; i++)
                {
                    if (testData[i].tur== testData[i].knnTur)
                    {
                        dogruSay++;
                    }
                }
                Console.Out.WriteLine("");
```

```
                Console.Out.WriteLine("");
                Console.Out.WriteLine("");
                Console.Out.WriteLine("Başarı oranı yüzde : "+ dogruSay/2);
                Console.Out.WriteLine("");
                Console.Out.WriteLine("");
                Console.Out.WriteLine("");
                Console.WriteLine("başka bir k değeri için de işlem yapmak ister misiniz ?(e/E - h/H):
");
                string durum = Console.ReadLine();
                if (durum == "h" || durum == "H")
                {
                    break;
                }
            }

        }

static void dataListOut(List<Data> set,int k =0)//List<Data> türünden listenin yazdırılması

        {
            bool durum = k != 2;// en son verilerin tamamı yazdırılırken hesaplanmış uzaklıkların
yazdırılmasına gerek yoktur.
            Console.Write(string.Format("{0,20:0.0}", "Varyans"));
            Console.Write(string.Format("{0,20:0.0}", "Çarpıklık"));
            Console.Write(string.Format("{0,20:0.0}", "Basıklık"));
            Console.Write(string.Format("{0,20:0.0}", "Entropi"));
            Console.Write(string.Format("{0,20:0.0}", "Tür"));
            if (durum)
            {
                Console.Write(string.Format("{0,20:0.0}", "Uzaklık"));
            }
            Console.Out.WriteLine("");
            for (int i = 0; i < set.Count; i++)
            {
                Console.Write(string.Format("{0,20}", set[i].varyans));
                Console.Write(string.Format("{0,20}", set[i].carpıklık));
                Console.Write(string.Format("{0,20}", set[i].basıklık));
                Console.Write(string.Format("{0,20}", set[i].entropi));
                Console.Write(string.Format("{0,20}", set[i].tur));
                if (durum) {
                    Console.Write(string.Format("{0,20:0.00000000}", set[i].distance));
                }
                Console.Out.WriteLine("");
            }
        static List<Data> KNNAlgo(Data dat, int k, List<Data> set)
        {

            MyList linkedList = new MyList() ;
            for (int i = 0; i < set.Count; i++)
            {
                set[i].distance = Math.Sqrt(Math.Pow(set[i].varyans - dat.varyans, 2) +
Math.Pow(set[i].carpıklık - dat.carpıklık, 2) + Math.Pow(set[i].basıklık - dat.basıklık, 2) +
Math.Pow(set[i].entropi - dat.entropi, 2));
                linkedList.add(set[i]);//sıralanarak ekleniyor araya item ekleneceği zaman bütün
itemlerin indexlerinin arttırılmasına gerek kalmıyor
            }
            return linkedList.returnList(k,dat);// girilen datanın knn ile türünün bulunması ve
Linkedlist yerine işlem kolaylığı için List<data> sınıfından bir liste döndürüyor


        }

  class MyList{
            public Node head;
            public int count=0;
            internal class Node
            {
                internal Data dat;
                internal Node prev;
                internal Node next;
                public Node(Data d)
                {
                    this.dat = d;
                    this.prev = null;
```

```
            this.next = null;
        }

    }

    public void add(Data d)
    {
        if (d!=null) {
            Node tempNode = head;
            Node data = new Node(d);
            if (head != null)
            {

                while(true)
                {
                    if (tempNode.dat.distance < d.distance)
                    {
                        if (tempNode.next != null)
                        {
                            tempNode = tempNode.next;//listenin dönülmesi
                        } else//adding to end of the list
                        {
                            count++;
                            tempNode.next = data;
                            data.prev = tempNode;
                            break;
                        }
                    }
                    else//doğru yer bulundu ekleme yapılması lazım
                    {
                        if (tempNode != head)//ilk elemandan sonra eklenme
                        {
                            Node current = tempNode;
                            current = current.prev;
                            data.prev = current;
                            data.next = tempNode;
                            current.next= data;
                            tempNode.prev = data;
                            break;
                        }
                        else//ilk elemanın önüne eklmek için
                        {
                            tempNode.prev = data;
                            data.next = tempNode;
                            head = data;
                            count++;
                            break;
                        }
                    }
                }

            }
            else//liste boşsa
            {
                head = data;
                count++;
            }
        }
       }

    public List<Data> returnList(int k,Data d)//listenin baştan k ya kadar olanının geri
döndürülmesi ve giriş datasının boğruluk değernin ortaya çıkartılması
    {
        List<Data> forReturn = new List<Data>(k);
        Node tempNode = head;
        double sum = 0;
        for(int i = 0; i < k && tempNode !=null; i++)
        {
            sum +=tempNode.dat.tur;
            forReturn.Add(tempNode.dat);
            tempNode=tempNode.next;
        };
        if (sum / k > 0.5)//eğer kendisine yakın değerlerde doğru sayısı fazla ise
        {
```

```
                d.knnTur = 1;
            }else if (sum / k < 0.5)
            {
                d.knnTur = 0;
            }
            else// eğer sahte ve doğru tür sayısı eşitse baştaki elemanın değerindedir.
            {
                d.knnTur = head.dat.tur;
            }
            return forReturn;
        }

        public void lListOut()//kontrol amaçlı link listin itemlerinin yazdırılması için
        {
            Node temp = head;
            while(temp!=null)
            {
                Console.Out.WriteLine(temp.dat.toString());
                temp = temp.next;
            }
        }


    }
```

2.c.2 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz

```
Lütfen bir k değeri giriniz:
2
K 2 için hesaplanıyor
Veri 0 asıl türü: 0 knn ile hesaplanan türü: 0
            Varyans          Çarpıklık          Basıklık          Entropi          Tür          Uzaklık
             2.8521              9.171           -3.6461           -1.2047            0       0.43108275
             2.8033             9.0862           -3.3668           -1.0224            0       0.55782895

Veri 1 asıl türü: 0 knn ile hesaplanan türü: 0
            Varyans          Çarpıklık          Basıklık          Entropi          Tür          Uzaklık
             3.8846            -3.0336            2.5334           0.20214            0       0.34597640
             4.1927            -3.2674            2.5839           0.21766            0       0.44667648

Veri 2 asıl türü: 0 knn ile hesaplanan türü: 0
            Varyans          Çarpıklık          Basıklık          Entropi          Tür          Uzaklık
```

200 veri içinde belirtilen k değeri kadar verinin çıkışı yapıldıktan sonra başarı oranı yazdırılmaktadır.

```
Veri 199 asıl türü: 1 knn ile hesaplanan türü: 1
            Varyans          Çarpıklık          Basıklık          Entropi          Tür
             -2.899           -0.60424             2.6045            1.3776            1
            -2.3898           -0.78427             3.0141           0.76205            1



Başarı oranı yüzde : 100
```

2.c.3 Açıklama

//Kullanılan veri yapıları ve algoritmanın kısaca anlatımını burada gerçekleştiriniz

Gerekli veri setleri olulturulduktan sonra girilen k değeri için knn metodu içerisine herbir test verisi veri seti göderilerek verinn türü tahmin ediliyor. Bu tahminden sonra Data nesnesinin knnTur adlı verisine işleniyor. Veri için gerçek türü ve tahminlenen türü karşılaştırılıp, k kadar verinin değerleri ve uzaklıkları yazdırılıyor.

Başarı oranı hesaplanırken knn metotundan sonra türler birbirlerine eşit ise başarılı tahmin değişkenine bir ekleniyor. 200 verinin işlemleri bittikten sonra başarı oranı (başarılı tahmin/200)*100 olarak yazdırılıyor.

H veya h girişi yapılana kadar bu veri setinde başka k değerleri içinde bu hesaplamaların hepsi tekrar tekrar yapılmaktadır.

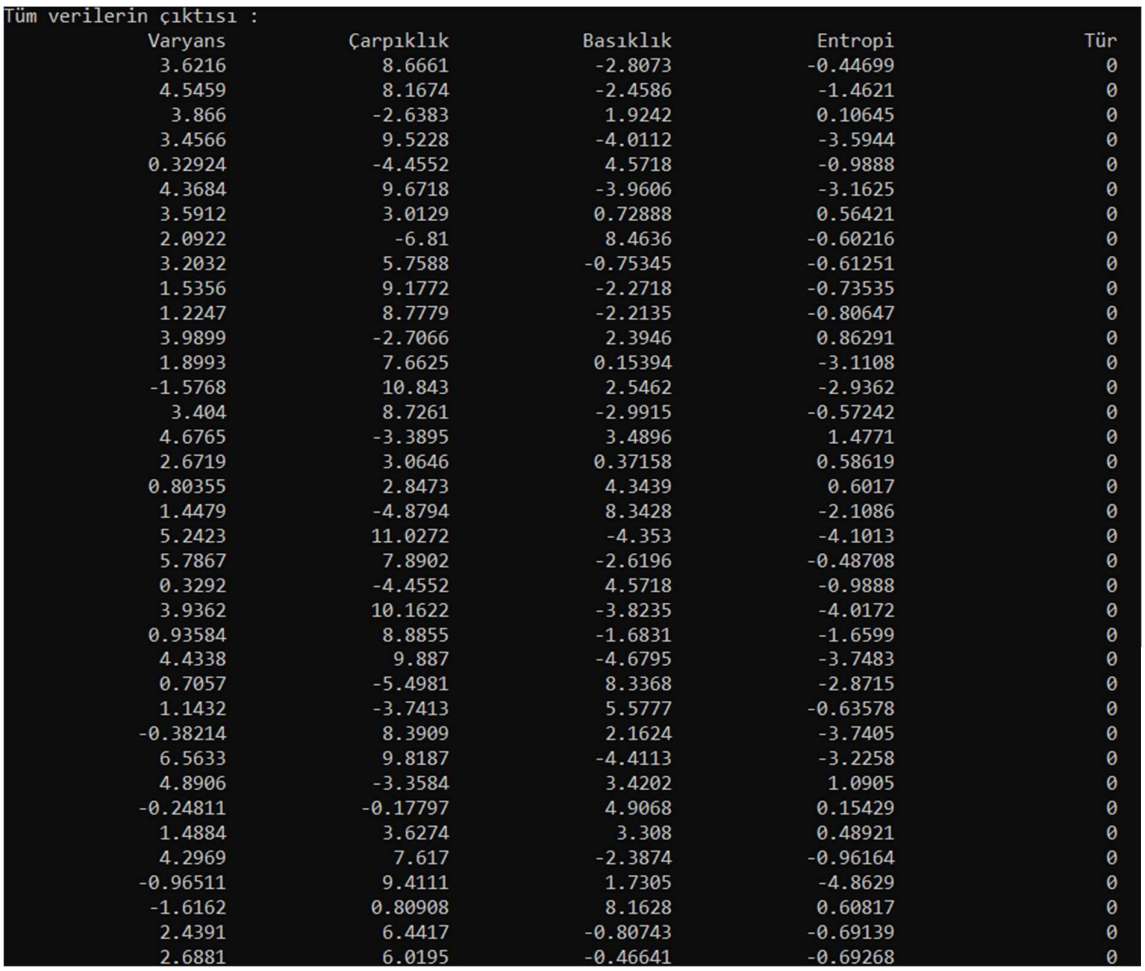
2.d Listeleme

2.d.1 Kodlar

```
        static void dataListOut(List<Data> set,int k =0)//List<Data> türünden listenin yazdırılması

        {
            bool durum = k != 2;// en son verilerin tamamı yazdırılırken hesaplanmış uzaklıkların
yazdırılmasına gerek yoktur.
            Console.Write(string.Format("{0,20:0.0}", "Varyans"));
            Console.Write(string.Format("{0,20:0.0}", "Çarpıklık"));
            Console.Write(string.Format("{0,20:0.0}", "Basıklık"));
            Console.Write(string.Format("{0,20:0.0}", "Entropi"));
            Console.Write(string.Format("{0,20:0.0}", "Tür"));
            if (durum)
            {
                Console.Write(string.Format("{0,20:0.0}", "Uzaklık"));
            }
            Console.Out.WriteLine("");
            for (int i = 0; i < set.Count; i++)
            {
                Console.Write(string.Format("{0,20}", set[i].varyans));
                Console.Write(string.Format("{0,20}", set[i].carpıklık));
                Console.Write(string.Format("{0,20}", set[i].basıklık));
                Console.Write(string.Format("{0,20}", set[i].entropi));
                Console.Write(string.Format("{0,20}", set[i].tur));
                if (durum) {
                    Console.Write(string.Format("{0,20:0.00000000}", set[i].distance));
                }
                Console.Out.WriteLine("");
            }
         }
```

2.d.2 Ekran görüntüleri



Tüm verilerin çıktısı :

Varyans	Çarpıklık	Basıklık	Entropi	Tür
3.6216	8.6661	-2.8073	-0.44699	0
4.5459	8.1674	-2.4586	-1.4621	0
3.866	-2.6383	1.9242	0.10645	0
3.4566	9.5228	-4.0112	-3.5944	0
0.32924	-4.4552	4.5718	-0.9888	0
4.3684	9.6718	-3.9606	-3.1625	0
3.5912	3.0129	0.72888	0.56421	0
2.0922	-6.81	8.4636	-0.60216	0
3.2032	5.7588	-0.75345	-0.61251	0
1.5356	9.1772	-2.2718	-0.73535	0
1.2247	8.7779	-2.2135	-0.80647	0
3.9899	-2.7066	2.3946	0.86291	0
1.8993	7.6625	0.15394	-3.1108	0
-1.5768	10.843	2.5462	-2.9362	0
3.404	8.7261	-2.9915	-0.57242	0
4.6765	-3.3895	3.4896	1.4771	0
2.6719	3.0646	0.37158	0.58619	0
0.80355	2.8473	4.3439	0.6017	0
1.4479	-4.8794	8.3428	-2.1086	0
5.2423	11.0272	-4.353	-4.1013	0
5.7867	7.8902	-2.6196	-0.48708	0
0.3292	-4.4552	4.5718	-0.9888	0
3.9362	10.1622	-3.8235	-4.0172	0
0.93584	8.8855	-1.6831	-1.6599	0
4.4338	9.887	-4.6795	-3.7483	0
0.7057	-5.4981	8.3368	-2.8715	0
1.1432	-3.7413	5.5777	-0.63578	0
-0.38214	8.3909	2.1624	-3.7405	0
6.5633	9.8187	-4.4113	-3.2258	0
4.8906	-3.3584	3.4202	1.0905	0
-0.24811	-0.17797	4.9068	0.15429	0
1.4884	3.6274	3.308	0.48921	0
4.2969	7.617	-2.3874	-0.96164	0
-0.96511	9.4111	1.7305	-4.8629	0
-1.6162	0.80908	8.1628	0.60817	0
2.4391	6.4417	-0.80743	-0.69139	0
2.6881	6.0195	-0.46641	-0.69268	0

1372 verinin tamamı yazdırılmaktadır.

2.d.3 Açıklama

Bellekte tutulan adlı veri lisitesi dataListOut metotuna parametre 2 verilir. dataListOut ise bu listedeki data nesnelerini eskiden hesaplanmış olan distance değerlerini yazdırmadan bütün verilerilerin çıktılarını yazdırır.

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	10	Yapıldı
Özdeğerlendirme Tablosu	10	10	Yapıldı

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