
***BLM 103- Algoritmalar ve Programlama I***
***2019-2020 Güz Dönemi***

# *Yapay Zeka ile Kanser Tespiti*

***Proje Teslim Raporu***
***14 Ocak 2020***

***Özlem ÇALI, Deniz Uzun***

*Öğretim Elemanı: Dr. Vecdi Emre Levent, emre.levent@fbu.edu.tr, İzinsiz Kopyalanamaz*

# 1 Giriş

## 1.1 Projenin Amacı

UC Irvine Üniversitesi'nin sağlamış olduğu Göğüs Kanseri verileri işlenerek yapay zeka uygulamalarında kullanılan bir algoritma ile kNN (k Nearest Neighborhood, En Yakın k Komşu) hasta olup olmadığı belli olmayan bir kişinin verileri sisteme beslenerek, hastalık tahmini yapılması sağlanmaya çalışılacaktır.

## 1.2 Proje Ekibi

Özlem ÇALI:
Okul numarası:190301002
Doğum Tarihi:19.05.2000
Doğum Yeri: Hatay
Mezun Olduğu Lise: Necmi Asfuroğlu Anadolu Lisesi

Deniz UZUN:
Okul numarası:190301015
Doğum Tarihi:08.04.2001
Doğum Yeri: İstanbul
Mezun Olduğu Lise: Kavacık Uğur Anadolu Lisesi

# 2 Geliştirilen Uygulama

## 2.1 Kullanılan Araçlar

Bu tasarım geliştirilirken Microsoft'un derleyicisi olan Visual Studio Community aracı kullanılacaktır..

**Visual Studio Community**
Visual Studio, birçok programlama dilini kullanarak program, uygulama ya da web sitesi yapabileceğiniz bir IDE yani entegre geliştirme ortamıdır. Microsoft Windows için bilgisayar programları, web siteleri, web uygulamaları, web hizmetleri ve mobil uygulamalar geliştirmek için kullanılır.

Visual Studio, Windows API, Windows Forms, Windows Presentation Foundation, Windows Store ve Microsoft Silverlight gibi Microsoft yazılım geliştirme platformlarını kullanır. Hem yerel kod hem de yönetilen kod üretebilir.

Visual Studio, IntelliSense'i (kod tamamlama bileşeni) ve kod yeniden düzenleme işlemini destekleyen bir kod düzenleyici içerir. Entegre hata ayıklayıcı, hem kaynak düzeyinde hata ayıklayıcı hem de makine düzeyinde hata ayıklayıcı olarak çalışır. Diğer yerleşik araçlar arasında bir kod profili oluşturucu, GUI uygulamaları oluşturmak için form tasarımcısı, web tasarımcısı, sınıf tasarımcısı ve veritabanı şeması tasarımcısı bulunur. Neredeyse her düzeyde işlevselliği artıran eklentileri kabul eder.

**Kullanılan veriler**

UC Irvine Üniversitesi veritabanından alınmış göğüs kanseri verileri

## 2.2 Tasarım

Tasarıma başlarken öncelikle kullanılacak olan kNN algoritması hakkında bilgi sahibi olduk.

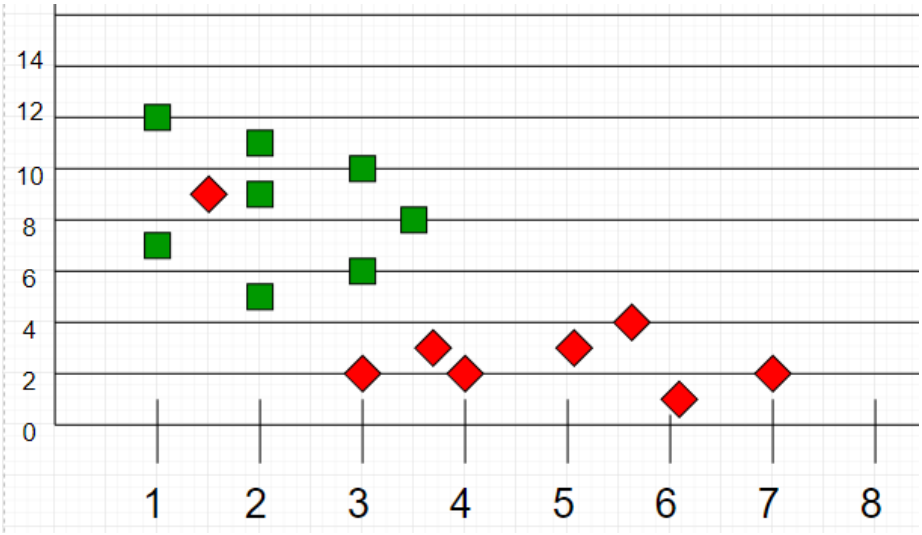
Knn algoritması; (sınıf niteliği belli olan) elemanların meydana getirdiği uzaya yeni bir örnek (sınıf niteliği belli olmayan) eklendiğinde bu örneğin kendisine en yakın olan sınıfa dahil edilmesi gerektiğini kararlaştıran bir algoritmadır. Kendisine en yakın olan sınıfı belirlemek için bir k değişkeni kullanmaktadır. Belirlenen bu k değişkeni örneğe en yakın olan k adet (sınıf nitelikleri belli olan) elemanların sayısını temsil etmektedir.
Örnek ile, elemanlar arasındaki mesafe hesaplanırken yaygın olarak aşağıdaki 3 uzaklık fonksiyonu kullanılmaktadır.Öklid fonksiyonu ilk okul veya liseden beri aşina olduğumuz bir fonksiyon. Minkowski fonksiyonundaki q=1 değeri için manhattan uzaklığı; q=2 değeri için öklid uzaklık formulünü elde ediyoruz. Bu sebeple minkowski genel formulunu kullanarak uygulamamızı yapacağız. Böylece kişi diğer uzaklık ölçümleri için sadece q değerini değiştirecektir.

kNN algoritması bir örneğin, veriseti içindeki kendisine uzaklık olarak en yakın olan k adet örneğin sınıf bilgisi en çok ne ise, sınıf bilgisi olarak bu değeri alır. Algoritmanın adımları aşağıda sıralanmaktadır.
1. En yakın k komşu sayısı için bir k değeri belirle, tek sayı olmalıdır. (3, 5, 7 … gibi)
2. Test edilecek veriyi, verisetindeki tüm verilere göre uzaklığını bul
3. Uzaklıkları küçükten büyüğe doğru sırala
4. Sıralanmış örneklerin en küçük uzaklığa sahip k örneği al
5. K örneğin içindeki sınıf değeri en çok ne ise, test edilen verinin sınıf değeri olarak belirle

K-En Yakın Komşular, Makine Öğreniminde en temel ancak temel sınıflandırma algoritmalarından biridir. Denetimli öğrenme alanına aittir ve örüntü tanıma, veri madenciliği ve saldırı tespitinde yoğun bir uygulama bulur.

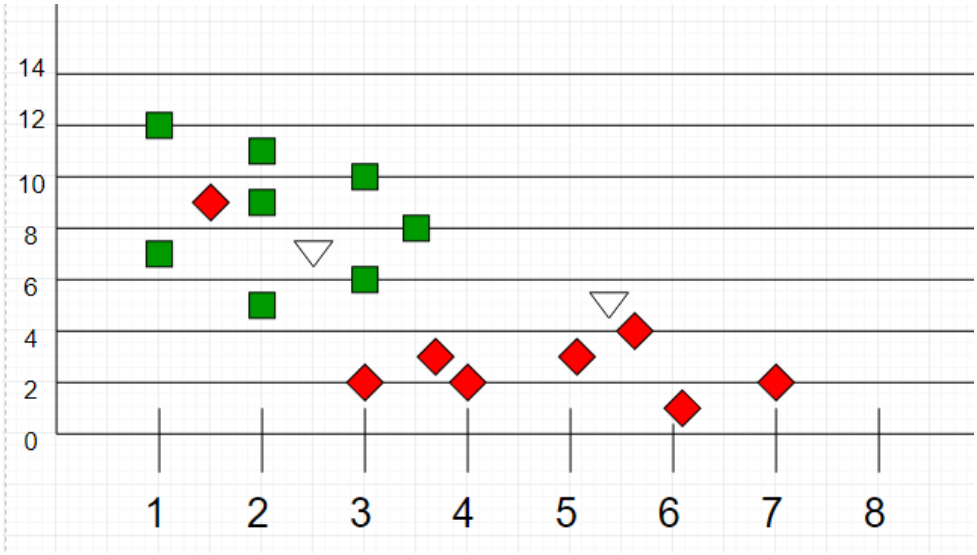
Parametrik olmadığı için gerçek hayat senaryolarında yaygın olarak tek kullanımlıktır, yani verilerin dağıtımı hakkında temel varsayımlar yapmaz ( verilen verilerin Gauss dağılımını varsayan GMM gibi diğer algoritmaların aksine ) .
Bize koordinatları bir özellik tarafından tanımlanan gruplara ayıran bazı önceki veriler (eğitim verileri de denir) verilir.

Örnek olarak, iki özellik içeren aşağıdaki veri noktaları tablosunu göz önünde bulundurun:




Şimdi, başka bir veri noktası seti (test verileri de denir) verildiğinde, eğitim setini analiz ederek bu noktaları bir gruba ayırın. Sınıflandırılmamış noktaların 'Beyaz' olarak işaretlendiğine dikkat edin.

,



**Sezgi**

Bu noktaları bir grafiğe çizersek, bazı kümeleri veya grupları bulabiliriz. Şimdi, sınıflandırılmamış bir nokta verildiğinde, en yakın komşularının hangi gruba ait olduğunu gözlemleyerek bir gruba atayabiliriz.
Bu, 'Kırmızı' olarak sınıflandırılan nokta kümesine yakın bir noktanın 'Kırmızı' olarak sınıflandırılma olasılığının daha yüksek olduğu anlamına gelir.
Sezgisel olarak, birinci noktanın (2.5, 7) 'Yeşil' ve ikinci noktanın (5.5, 4.5) 'Kırmızı' olarak sınıflandırılması gerektiğini görebiliriz.

**Algoritma**

Eğitim verileri örnek sayısı m olsun. P bilinmeyen bir nokta olsun.

1. Antrenman örneklerini bir dizi veri noktası dizisinde saklayın .Bu, bu dizideki her öğenin bir tupleı (x, y) temsil ettiği anlamına gelir.
2. i = 0 ila m için:
3. Öklid mesafesini d (dizi [i], p) hesaplayın.
4. K kümesini elde edilen en küçük mesafeler yapın. Bu mesafelerin her biri önceden sınıflandırılmış bir veri noktasına karşılık gelir.
5. Çoğunluk etiketini S arasında iade edin.

K, tek bir sayı olarak tutulabilir, böylece sadece iki grubun mümkün olduğu durumlarda (örneğin, Kırmızı / Mavi) açık bir çoğunluğu hesaplayabiliriz. Artan K ile, farklı sınıflandırmalarda daha pürüzsüz, daha tanımlanmış sınırlar elde ederiz. Ayrıca, eğitim setindeki veri noktalarının sayısını artırdığımız için yukarıdaki sınıflandırıcının doğruluğu da artar.

k-NN bu kadar etkin ve kolay bir algoritma olmasına karşın bazı önemli dezavantajlara sahiptir.

- Örnek sayısı arttığında yapmamız gereken karşılaştırma işlemlerinin sayısı da lineer bir şekilde artmaktadır ve bu çok ağır bir işlem yükü getirmektedir.
- Özellikle boyut sayısı ve k sayısı arttığında kNN algoritması "overfitting" problemi ile karşı karşıya kalmaktadır.

## Yapılma aşamaları

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

int kullanicigirisihesapla(int diziegitim[600][10]);
int sonucbul(int diziegitim[600][10],int dizitest[83][10]);

int main()
{

    FILE* dosya;
    FILE* dosya2;
    int i;
    int istenilen;
    int a;
    int diziegitim[600][10];
    int dizitest[83][10];
    int sayi1,sayi2,sayi3,sayi4,sayi5,sayi6,sayi7,sayi8,sayi9,sayi10;
    int sayi11,sayi12,sayi13,sayi14,sayi15,sayi16,sayi17,sayi18,sayi19,sayi20;
```

Öncelikle yaptığımız kod, fonksiyon şeklinde olduğu için bu fonksiyonları tanımladık.Bize verilen eğitim ve test verilerini dosyadan okumak için gerekli tanımlamaları yaptık.Her bir verisetinde 11 tane sayı bulunmaktadır.Bu 11 sayıdan ilki hastadan alınan örneğin numarası iken, son sayı sınıf bilgisi yani hasta olup olmadığını bilgilendirmektedir. Geriye kalan 9 sayı hastanın değerleridir.Eğitim verisetinde 600 örnek varken test verisetinde 83 tane örnek olduğu için dizi halinde belirttik.

,

```c
fopen_s(&dosya, "egitim.txt", "r");


for(i=0;i<600;i++) {

fscanf_s(dosya,"%*d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d",&sayi1,&sayi2,&sayi3,&sayi4,&sayi5,&sayi6,&sayi7,&sayi8,&sayi9,&sayi10);

diziegitim[i][0]=sayi1;
diziegitim[i][1]=sayi2;
diziegitim[i][1]=sayi2;
diziegitim[i][2]=sayi3;
diziegitim[i][3]=sayi4;
diziegitim[i][4]=sayi5;
diziegitim[i][5]=sayi6;
diziegitim[i][6]=sayi7;
diziegitim[i][7]=sayi8;
diziegitim[i][8]=sayi9;
diziegitim[i][9]=sayi10;


}
fclose(dosya);
```

Dizi halinde eğitim veriseti dosyadan okutulmaktadır.

```
fopen_s(&dosya2,"test.txt","r");
for(a=0;a<83;a++) {

fscanf_s(dosya,"%*d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d",&sayi11,&sayi12,&sayi13,&sayi14,&sayi15,&sayi16,&sayi17,&sayi18,&sayi19,&sayi20);

dizitest[a][0]=sayi11;
dizitest[a][1]=sayi12;
dizitest[a][2]=sayi13;
dizitest[a][3]=sayi14;
dizitest[a][4]=sayi15;
dizitest[a][5]=sayi16;
dizitest[a][6]=sayi17;
dizitest[a][7]=sayi18;
dizitest[a][8]=sayi19;
dizitest[a][9]=sayi20;

}


fclose(dosya2);
```

Dizi halinde test veriseti dosyadan okutulmaktadır.

```
home:
printf("Test dosyasindaki verileri hesaplamak icin 1 , Kullanicidan parametre alip verileri hesaplamak icin 2 giriniz:");
scanf_s("%d",&istenilen);
if(istenilen==1){

sonucbul(diziegitim,dizitest);

}

else if(istenilen==2){

kullanicigirisihesapla(diziegitim);
}
else {
    printf("HATALI GİRİŞ TEKRAR GİRİN:");
 goto home;
}
```

Kod, fonksiyon şeklinde olduğu için öncelikle kullanıcıya ne  yapmak istediği sorulmaktadır.

```
int sonucbul(int diziegitim[600][10],int dizitest[83][10]) {
    int x,i,j;
    int m;
    int z;
    int sonuc=0,hastalikdegeri,enkucuk,gecicibellek,gecicibellek2;
    int hasta;
    int hastadegil;
    int dizitestsonucu[600][2];
    int dizitestsonucuDY[6];
     char dizitestsonucuDYsay[83];
    int k=5;
    float dogrusayisi=0;
    float yanlissayisi=0;
```

Kullanıcı 1’e bastıysa sonucbul fonksiyonuna gidilmektedir. Eğitim ve test dizileri,bu kısmı hesaplamada gerekli olan diğer argümanlar tanımlanıyor.

```
for(x=0;x<83;x++)
   {
       for (j=0;j<600;j++)
       {
           sonuc=0;
           for(z=0;z<9;z++)
           {
               int diff = (diziegitim[j][z]-dizitest[x][z]);

               sonuc=sonuc+ diff * diff;

           dizitestsonucu [j][0]=sonuc;
           }

           hastalikdegeri=diziegitim[j][9];
           dizitestsonucu[j][1]=hastalikdegeri;
       }
```

Test ve eğitim verisetinin hasta numarası ve sınıf bilsi hariç diğer 9 değerler arasındaki uzaklıkların bulunması için bir döngü kurduk. Test verisetindeki her bir hasta eğitim verisetinde 600 hastaya olan uzaklıkları bulunuyor.
Sonuçlar iki boyutlu dizide kaydedilmektedir.
Dizinin 0. İndeksine uzaklık değeri , 1.indeksine sınıf bilgisi kaydedilmektedir.

```
 for (i = 0; i < 599; i++)
{
enkucuk= i;
for (j = i + 1; j < 600; j++){

if (dizitestsonucu[j][0] <dizitestsonucu[enkucuk][0])

    enkucuk= j;
}
gecicibellek=dizitestsonucu [enkucuk][0];
dizitestsonucu[enkucuk][0] = dizitestsonucu[i][0];
dizitestsonucu[i][0] = gecicibellek;

gecicibellek=dizitestsonucu [enkucuk][1];
dizitestsonucu[enkucuk][1] = dizitestsonucu[i][1];
dizitestsonucu[i][1] = gecicibellek;

}
```

Selection sord algoritması ile bulduğumuz uzaklıkları sıraladık.Sonuçları dizitestsonuc dizisinin 0.indeksine bağlı olarak sıraladık.

```
hasta=0;
hastadegil=0;
    for (i=0;i<k;i++)
    {
        gecicibellek2=dizitestsonucu[i][1];
         if (gecicibellek2==2) {
             hastadegil++;
         }
         else {
             hasta++;
         }
    }

    if (hastadegil>hasta) {
        dizitestsonucuDY[i]=2;
    }
    else  {
        dizitestsonucuDY[i]=4;
    }

    if (dizitestsonucuDY[i]==dizitest[x][9])  {
        dizitestsonucuDYsay[x]='D';
    }

    else  {
        dizitestsonucuDYsay[x]='Y';
    }

}
```

En yakın seçtiğimiz k tane dizitestsonucu değerinin 1.indeksine göre hasta ise 4'ü,hasta değil ise 2'yi arttırdık. Eğer hasta değiller fazla ise dizittestsonucuDY dizisine 2 ,eğer hastalar fazla ise 4 yazdırdık.
Bu diziyi en son dizitest ile karşılaştırdık. Eğer eşleşiyorlarsa dizitestsonucuDYsay 'a 'D', eşleşmiyorlarsa 'Y' yazdırdık.

```
for(m=0;m<83;m++)
{
    printf("%d.veri== %c\n",m+1,dizitestsonucuDYsay[m]);
    if (dizitestsonucuDYsay[m]=='D') {
        dogrusayisi++;
    }
    else  {
        yanlissayisi++;
    }
}

printf("\n DOĞRU SAYISI:%f\n YANLIŞ SAYISI:%f\n",dogrusayisi,yanlissayisi);

printf("BAŞARI ORANI YÜZDE:%.3f\n",((dogrusayisi*100)/83));
getchar();
getchar();
return 0;
}
```

Yaptığımız 83 tane test verisetinde kaç tane doğru kaç tane Y olduğunu hesaplattırdık.
Son olarak başarı oranını hesapladık.

```
int kullanicigirisihesapla(int diziegitim[600][10])
{
    int dizikullanicigirisi[9];
    int dizikullanicigirisiuzaklik[600][2];
int gecicibellek,gecicibellek2,sonuc,hastalikdegeri,i,j,enkucuk;
int k=5,hasta=0,hastadegil=0;
int z;


for(i=0;i<9;i++){
    printf("%d.degeri giriniz:",i+1);
    scanf_s("%d",&gecicibellek);
    dizikullanicigirisi[i]=gecicibellek;
}
```

Kullanıcıdan parametre alıp hastanın kanser olup olmadığını hesaplamak için gerekli argümanları tanımladık.
dizikullanicigirisi adında bir dizi tanımlayarak 9 değeri kullanıcıdan aldık.

```
for (j=0;j<600;j++) {

    sonuc=0;
    for(z=0;z<9;z++) {
        int diff = (diziegitim[j][z]-dizikullanicigirisi[z]);

                sonuc=sonuc+ diff * diff;
                hastalikdegeri=diziegitim[j][9];

            dizikullanicigirisiuzaklik[j][0]=sonuc;
            dizikullanicigirisiuzaklik[j][1]=hastalikdegeri;
    }
}
```

Aldığımız 9 değeri eğitim verisetindeki değerlere uzaklığını bulup ,bulunan sonucu dizikullanicigirisiuzaklik adındaki dizinin 0.indeksine kaydettik.
Sınıf bilgisini ise dizikullanicigirisiuzaklik adındaki dizinin 1.indeksine kaydettik.

```
for (i=0;i<599;i++) {
    enkucuk=i;
    for(j=i+1;j<600;j++)
    {
        if   (dizikullanicigirisiuzaklik[j][0]<dizikullanicigirisiuzaklik[enkucuk][0])
            enkucuk=j;

    gecicibellek=dizikullanicigirisiuzaklik[enkucuk][0];
    dizikullanicigirisiuzaklik[enkucuk][0]=dizikullanicigirisiuzaklik[i][0];
    dizikullanicigirisiuzaklik[i][0]=gecicibellek;

    gecicibellek=dizikullanicigirisiuzaklik[enkucuk][1];
    dizikullanicigirisiuzaklik[enkucuk][1]=dizikullanicigirisiuzaklik[i][1];
    dizikullanicigirisiuzaklik[i][1]=gecicibellek;
    }
}
```

Selection sord algoritması ile bulduğumuz uzaklık değerlerini sıraladık.

```
    for(i=0;i<k;i++) {
        gecicibellek2=dizikullanicigirisiuzaklik[i][1];
        if(gecicibellek2==2) {
            hastadegil++;
        }
        else {
            hasta++;
        }
    }

    if(hastadegil>hasta){
        printf("\n HASTA DEGİLSİNİZ");
    }
    else {
        printf("\n HASTASİNİZ");
    }

    getchar();
    getchar();
    return 0;
}
```

En yakın k tane değere göre hasta değil ve hasta değerlerini hesapladık.Bu değerlere göre eğer hasta sayısı hasta değil sayısından fazla ise ekrana HASTASINIZ yazısını,hasta değil sayısı fazla ise HASTA DEĞİLSİNİZ yazısını yazdırdık.

# 3 Sonuçlar

Yapay Zeka ile Kanser Tespiti projesini yaparken en önemlisi kNN algoritması ve Selection Sord algoritmaları hakkında bilgi sahibi olduk.

Bir çok sayıda bilgi varken o bilgileri dizi şeklinde ifade edip tanımlayabileceğimizi,dosya okuyup kapatmayı,dosya içerisindeki verileri dizi şeklinde tanımlayıp işlem yapabilmek gibi daha bir çok konu başlığına hakim olup bu projede aktif olarak kullandık.

Ayrıca bu proje ile günlerce hatta haftalarca bir kod üzerinde düşünmenin ,uğraşıp geliştirmenin deneyimini yaşadık.

Hazırlanan sunum video'su adresi: https://www.youtube.com/watch?v=bKQm0-GV7e8

Dosyaların github adresi: https://github.com/ozlemcali/Yapay-Zeka-le-Kanser-Tespiti